Mathematics of Neural Networks

Volume II: Probabilistic and Generative Models

Bayesian inference, latent variables, and the mathematics of generation

---

Every derivation step by step

Every symbol defined

Explicit numerical examples

Geometric intuitions

Silvano Salvador

Contents

13.1 Probability Spaces and Measures 3

13.2 Expectation and Key Inequalities 4

13.3 KL Divergence 6

13.4 Conditional Probability and Bayes' Theorem 8

13.5 Information Theory Foundations 10

13.6 Sufficient Statistics and the Fisher Information 11

14.1 The Exponential Family 14

14.2 Directed Graphical Models (Bayesian Networks) 15

14.3 Undirected Graphical Models (Markov Random Fields) 17

14.4 Hierarchical Bayesian Models 18

15.1 The Log-Likelihood Decomposition 21

15.2 The E-Step and M-Step 22

15.3 EM for Gaussian Mixture Models — Complete Derivation 24

16.1 The Evidence Lower BOund (ELBO) 27

16.2 Mean-Field Variational Inference 29

16.3 CAVI: Coordinate Ascent Variational Inference 30

16.4 Black-Box Variational Inference 32

16.5 Amortised Inference and the Inference Network 34

17.1 The Generative Model 35

17.2 The Inference Network and ELBO 37

17.3 The Reparameterisation Trick 39

17.4 Complete VAE Training Step 41

18.1 The Change-of-Variables Formula 43

18.2 Autoregressive Flows 45

18.3 Coupling Layers and RealNVP 47

18.4 Glow: Generative Flow with Invertible 1x1 Convolutions 49

19.1 The Minimax Objective 51

19.2 Wasserstein GAN 53
19.3 Training Dynamics and Mode Collapse 54
19.4 Spectral Normalisation for Discriminators 56
20.1 The Forward Process 58
20.2 The Reverse Process and ELBO 59
20.3 The DDPM Training Objective 61
20.4 The Probability Flow ODE 63
20.5 Classifier-Free Guidance — Complete Derivation 65
20.6 Score-Based Generative Models 66
20.7 DDIM: Deterministic Sampling 68
21.1 Continuous Normalising Flows and the Continuity Equation 70
21.2 Conditional Flow Matching Objective 72
21.3 Optimal Transport CFM (OT-CFM) 74
22.1 Energy-Based Models 76
22.2 Score Matching 78
22.3 Contrastive Divergence Training 79
22.4 Noise Contrastive Estimation (NCE) 81
23.1 Gaussian Processes 83
23.2 Deep Gaussian Processes 85
23.3 GP Regression: Full Derivation 87
23.4 Kernel Functions and Gaussian Process Priors 88
24.1 Comparison of Generative Frameworks 92
24.2 Theoretical Connections 93
24.3 Summary of Volume II 96
Markov Chain Monte Carlo and Langevin Dynamics 100
Elements of Measure-Theoretic Probability 103
Stochastic Optimisation and Gradient Estimation 105

PART 13

Probability Theory: Measure-Theoretic Foundations

We lay the rigorous probabilistic foundations needed for generative modelling: measure theory, random variables, densities, and the key inequalities that underpin variational inference.

13.1 Probability Spaces and Measures

Independence and correlation.

$$X \perp Y \Leftrightarrow P(X \in A, Y \in B) = P(X \in A)P(Y \in B) \quad \forall A, B$$

$$\mathrm{Cov}(X,Y) = \mathbb{E}[(X - \mathbb{E}X)(Y - \mathbb{E}Y)], \quad \rho_{XY} = \frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)}} \in [-1,1]$$

L'indipendenza implica correlazione nulla (ma non viceversa).

Random variables and distributions. A *random variable* $X : \Omega \to \mathbb{R}$ is a measurable function from the sample space to the reals. Its *distribution* (or *law*) is the push-forward measure $P_X = P \circ X^{-1}$: for Borel set $B \subseteq \mathbb{R}$, $P_X(B) = P(X^{-1}(B))$.

For a *continuous* random variable with density f: $P(a \le X \le b) = \int_a^b f(x)\,dx$. The density satisfies $f \ge 0$ and $\int_{-\infty}^{\infty} f(x)\,dx = 1$.

Why σ-algebras matter for deep learning.

- *Conditional expectations:* $\mathbb{E}[Y|\mathcal{F}]$ is defined as the $\mathcal{F}$-measurable random variable satisfying $\int_A \mathbb{E}[Y|\mathcal{F}]\,dP = \int_A Y\,dP$ for all $A \in \mathcal{F}$. This generalises $\mathbb{E}[Y|X = x]$ and underpins the theory of VAEs and flow matching.
- *Independence:* X and Y are independent iff $\sigma(X)$ and $\sigma(Y)$ are independent σ-algebras, i.e. $P(A \cap B) = P(A)P(B)$ for all $A \in \sigma(X)$, $B \in \sigma(Y)$.
- *Measurability of neural networks:* a composition of measurable functions is measurable, so neural networks define valid random variables when inputs are random.

Key distributions in deep learning.

Distribution	Density / PMF	Mean	Variance
Bernoulli(p)	$p^x(1-p)^{1-x}$	p	$p(1-p)$
Categorical$(p_{1:K})$	$\prod_k p_k^{[x=k]}$	—	—
$\mathcal{N}(\mu, \sigma^2)$	$(2\pi\sigma^2)^{-1/2} e^{-(x-\mu)^2/2\sigma^2}$	μ	σ^2
$\mathcal{N}(\square, \square)$	$(2\pi)^{-d/2}\lvert\square\rvert^{-1/2} e^{-\frac{1}{2}(\mathbf{x}-\square)^\top \square^{-1}(\mathbf{x}-\square)}$	$\square$	$\square$
Laplace(μ, b)	$\frac{1}{2b} e^{-\lvert x-\mu\rvert/b}$	μ	$2b^2$
Beta(α, β)	$x^{\alpha-1}(1-x)^{\beta-1}/B(\alpha,\beta)$	$\frac{\alpha}{\alpha+\beta}$	—

The multivariate Gaussian is the base distribution of VAEs, normalising flows, and diffusion models. The Laplace prior corresponds to L1/Lasso regularisation.

Constructing σ-algebras. The smallest σ-algebra containing a collection $\mathcal{C}$ of subsets is $\sigma(\mathcal{C})$, the σ-algebra *generated* by $\mathcal{C}$. For $\Omega = \mathbb{R}^d$, the Borel σ-algebra $\mathcal{B}(\mathbb{R}^d) = \sigma(\{\text{open sets}\})$ is the standard choice. Every open set, closed set, countable union or intersection of open/closed sets is Borel-measurable.

Lebesgue measure. On $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$, the Lebesgue measure λ satisfies $\lambda([a_1, b_1] \times \cdots \times [a_d, b_d]) = \prod_{i=1}^{d}(b_i - a_i)$. It is σ-finite, translation-invariant, and unique (up to scaling) among such measures. Probability measures absolutely continuous with respect to Lebesgue measure have densities: $P(A) = \int_A p(\mathbf{x})\, d\lambda(\mathbf{x})$ with $p(\mathbf{x}) \geq 0$ and $\int p = 1$.

Radon-Nikodym theorem. If $P \ll Q$ (P is absolutely continuous w.r.t. Q, meaning $Q(A) = 0 \implies P(A) = 0$), there exists a $\mathcal{F}$-measurable function $f = dP/dQ \geq 0$ such that $P(A) = \int_A f\, dQ$ for all $A \in \mathcal{F}$. The function f is the *Radon-Nikodym derivative* or density of P with respect to Q. In ML: $p(\mathbf{x}) = dP/d\lambda$ is the probability density function when $P \ll \lambda$.

Random variables and pushforward measures. A random variable $X : \Omega \to \mathcal{X}$ induces a *pushforward measure* $P_X = P \circ X^{-1}$ on $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$: $P_X(B) = P(X^{-1}(B)) = P(\{\omega : X(\omega) \in B\})$. The distribution of X is P_X. For $X : \Omega \to \mathbb{R}^d$ with density p_X, the CDF is $F(\mathbf{x}) = P_X((-\infty, x_1] \times \cdots \times (-\infty, x_d]) = \int_{(-\infty, \mathbf{x}]} p_X(\mathbf{u})\, d\mathbf{u}$.

Example — Borel sets and measurability in practice

In deep learning, we rarely work with the formal probability space $(\Omega, \mathcal{F}, P)$ explicitly. Instead, we work with the induced distribution P_X on data space $\mathcal{X} = \mathbb{R}^d$. When we write $\mathbf{x} \sim p_{\text{data}}$, we mean sampling from the pushforward measure $P_X = p_{\text{data}}\, d\lambda$ where λ is the Lebesgue measure on $\mathbb{R}^d$. The assumption that p_{data} is a density (w.r.t. Lebesgue) rules out point masses; in practice, real images lie on a low-dimensional manifold, so p_{data} is technically singular. This is why diffusion models add noise — to regularise the distribution and make it absolutely continuous.

Independence and product spaces. $X_1, \ldots, X_n$ are *independent* iff the joint equals the product of marginals: $P_{X_1,\ldots,X_n} = P_{X_1} \otimes \cdots \otimes P_{X_n}$, i.e. $P(X_1 \in B_1, \ldots, X_n \in B_n) = \prod_{i=1}^{n} P(X_i \in B_i)$ for all $B_i \in \mathcal{B}(\mathcal{X})$. For densities: $p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_i p_i(\mathbf{x}_i)$. The i.i.d. assumption in ML: each training sample $(\mathbf{x}^{(i)}, y^{(i)})$ is drawn independently from the same P_{data}, giving $\log P(\mathcal{D}|\theta) = \sum_i \log p_\theta(\mathbf{x}^{(i)}, y^{(i)})$ (the factorisation underlying MLE).

13.2 Expectation and Key Inequalities

Expectation.

$$\mathbb{E}_p[f(X)] = \int_{\mathbb{R}^d} f(\mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}$$

Jensen's inequality. For a convex function ϕ:

$$\phi(\mathbb{E}[X]) \leq \mathbb{E}[\phi(X)]$$

Equality holds iff X is almost surely constant, or ϕ is linear.

Application. $-\log$ is convex, so $-\log(\mathbb{E}[X]) \leq \mathbb{E}[-\log(X)]$. This underpins the ELBO derivation (Part 16).

Markov's inequality. For $X \geq 0$ and $a > 0$:

$$P(X \geq a) \leq \frac{\mathbb{E}[X]}{a}$$

Moment generating function and characteristic function.

$$M_X(t) = \mathbb{E}[e^{tX}], \quad \varphi_X(t) = \mathbb{E}[e^{itX}] \quad (i = \sqrt{-1})$$

MGF characterises the distribution when it exists: $\mathbb{E}[X^k] = M_X^{(k)}(0)$. For $X \sim \mathcal{N}(\mu, \sigma^2)$: $M_X(t) = \exp(\mu t + \sigma^2 t^2/2)$.

Law of total expectation and variance.

$$\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]], \qquad \text{Var}(X) = \mathbb{E}[\text{Var}(X|Y)] + \text{Var}(\mathbb{E}[X|Y])$$

The variance decomposition (law of total variance) decomposes variance into within-group variance ("aleatoric") and between-group variance ("epistemic"). In Bayesian deep learning, aleatoric uncertainty is irreducible noise in the data; epistemic uncertainty comes from parameter uncertainty and decreases with more data.

Jensen's inequality in deep learning. For convex f: $f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$.

- *ELBO derivation:* $\log p(\mathbf{x}) = \log \mathbb{E}_{q(\mathbf{z})}[p(\mathbf{x}, \mathbf{z})/q(\mathbf{z})] \geq \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z})/q(\mathbf{z})] = \mathcal{L}_{\text{ELBO}}$ (by Jensen's with concave log).
- *Cross-entropy lower bound:* $H(p, q) \geq H(p)$ (entropy is concave, KL is non-negative).
- *Softmax temperature:* at $T \to 0$, $\text{softmax}(\mathbf{z}/T) \to$ one-hot$(\arg\max \mathbf{z})$. The log-sum-exp $\log \sum_k \exp(z_k)$ is convex (LSE approximates $\max_k z_k$).

Markov and Chebyshev inequalities. For non-negative X: $P(X \geq a) \leq \mathbb{E}[X]/a$. Chebyshev: $P(|X - \mu| \geq k\sigma) \leq 1/k^2$. These bound tail probabilities without distributional assumptions — fundamental in PAC learning and generalisation theory.

Law of Large Numbers. If $X_1, X_2, \ldots$ are i.i.d. with $\mathbb{E}[|X_1|] < \infty$, the Strong LLN states: $\bar{X}_n = \frac{1}{n}\sum_{i=1}^n X_i \xrightarrow{a.s.} \mathbb{E}[X_1]$. This justifies Monte Carlo estimation: for any function f and $\mathbf{z}^{(s)} \sim q$,

$$\hat{I}_S = \frac{1}{S}\sum_{s=1}^{S} f(\mathbf{z}^{(s)}) \xrightarrow{a.s.} \mathbb{E}_q[f(\mathbf{z})] \quad \text{as } S \to \infty$$

Central Limit Theorem. Under the same conditions with additionally $\text{Var}(X_1) = \sigma^2 < \infty$:

$$\sqrt{n}(\bar{X}_n - \mu) \xrightarrow{d} \mathcal{N}(0, \sigma^2)$$

The CLT gives the rate of convergence: the MC estimator's error is $O(1/\sqrt{S})$ regardless of dimension d. This dimension-independence is the key advantage of MC over deterministic quadrature (which suffers the curse of dimensionality).

Sub-Gaussian random variables. X is σ-sub-Gaussian if $\mathbb{E}[\exp(\lambda(X - \mathbb{E}[X]))] \leq \exp(\lambda^2\sigma^2/2)$ for all $\lambda \in \mathbb{R}$. Sub-Gaussian variables satisfy Hoeffding-type tail bounds:

$$P(|\bar{X}_n - \mu| \geq t) \leq 2\exp\left(-\frac{nt^2}{2\sigma^2}\right)$$

Gaussian(μ, σ^2) and bounded $[a, b]$ random variables are $(b-a)/2$-sub-Gaussian. Sub-Gaussianity is the standard assumption in PAC-learning bounds and generalisation theory.

Azuma-Hoeffding inequality. For a martingale difference sequence $\{D_i\}$ with $|D_i| \leq c_i$:

$$P\left(\left|\sum_{i=1}^n D_i\right| \geq t\right) \leq 2\exp\left(-\frac{2t^2}{\sum_i c_i^2}\right)$$

Used in learning theory to show that empirical risk concentrates around the true risk: $|\hat{R}(h) - R(h)| \leq O(1/\sqrt{n})$ with high probability.

13.3 KL Divergence

The Kullback-Leibler divergence is the most widely used measure of "distance" between distributions in deep learning: it appears in the cross-entropy loss, the VAE ELBO, knowledge distillation, and KL-regularised reinforcement learning.

KL Divergence

For two distributions P and Q over $\mathcal{X}$:

$$D_{\mathrm{KL}}(P\|Q) = \mathbb{E}_{x\sim P}\left[\log\frac{P(x)}{Q(x)}\right] = \sum_x P(x)\log\frac{P(x)}{Q(x)}$$

(discrete case; for continuous distributions: $\int p(x)\log\frac{p(x)}{q(x)}dx$).

Fundamental properties.

1. $D_{\mathrm{KL}}(P\|Q) \geq 0$, with equality iff $P = Q$ a.e. (*Gibbs' inequality*).
2. $D_{\mathrm{KL}}(P\|Q) \neq D_{\mathrm{KL}}(Q\|P)$: not symmetric.
3. If $\exists x : Q(x) = 0$ and $P(x) > 0$, then $D_{\mathrm{KL}}(P\|Q) = +\infty$.
4. Does not satisfy the triangle inequality: not a metric.

Proof of non-negativity (Jensen's inequality).

$$-D_{\mathrm{KL}}(P\|Q) = \mathbb{E}_P\left[\log\frac{Q(x)}{P(x)}\right] \leq \log\mathbb{E}_P\left[\frac{Q(x)}{P(x)}\right] = \log\sum_x P(x)\frac{Q(x)}{P(x)} = \log 1 = 0$$

The first inequality uses the concavity of log via Jensen.

Connection with cross-entropy.

$$D_{\mathrm{KL}}(P\|Q) = H(P,Q) - H(P)$$

where $H(P,Q) = -\mathbb{E}_P[\log Q(x)]$ is the cross-entropy and $H(P) = -\mathbb{E}_P[\log P(x)]$ is the entropy of P. Minimising cross-entropy (as in classification) is equivalent to minimising the KL divergence when P is the fixed target distribution.

Example — KL between Gaussians

$P = \mathcal{N}(\mu_1, \sigma_1^2)$, $Q = \mathcal{N}(\mu_2, \sigma_2^2)$:

$$D_{\mathrm{KL}}(P\|Q) = \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$$

Special case $P = \mathcal{N}(\mu, \sigma^2)$, $Q = \mathcal{N}(0, 1)$ (used in the VAE):

$$D_{\mathrm{KL}}(P\|Q) = \frac{1}{2}\left(\mu^2 + \sigma^2 - 1 - \log\sigma^2\right)$$

Forward KL vs. Reverse KL (mode-averaging vs. mode-seeking).

- **Forward KL** $D_{\mathrm{KL}}(p_{\text{data}}\|p_\theta)$: penalises regions where $p_\theta \approx 0$ but $p_{\text{data}} > 0$. Produces models that "cover" the full data distribution (mode-averaging). Used in MLE.

- **Reverse KL** $D_{\mathrm{KL}}(p_\theta \| p_{\text{data}})$: penalises regions where $p_\theta > 0$ but $p_{\text{data}} \approx 0$. Produces models focused on a subset of modes (mode-seeking). Used in RL (RLHF, PPO-clip) and variational inference.

KL divergence in variational inference. The ELBO decomposition (§16.1) follows directly:

$$\log p(\mathbf{x}) = \mathcal{L}_{\text{ELBO}}(q) + D_{\mathrm{KL}}(q(\mathbf{z}) \,\|\, p(\mathbf{z}|\mathbf{x})) \geq \mathcal{L}_{\text{ELBO}}(q)$$

since $D_{\mathrm{KL}}(\geq \| 0)$. Minimising $D_{\mathrm{KL}}(q(\mathbf{z}) \,\|\, p(\mathbf{z}|\mathbf{x}))$ over q is equivalent to maximising the ELBO — the fundamental variational principle.

Forward vs. reverse KL.

- *Forward KL:* $D_{\mathrm{KL}}(p \,\|\, q) = \mathbb{E}_p[\log p/q]$. Zero-avoiding: if $p(\mathbf{z}) > 0$ then $q(\mathbf{z}) > 0$ is required. Results in *mean-seeking* approximations that spread mass to cover all modes of p.
- *Reverse KL:* $D_{\mathrm{KL}}(q \,\|\, p) = \mathbb{E}_q[\log q/p]$. Zero-forcing: if $q(\mathbf{z}) > 0$ then $p(\mathbf{z}) > 0$ required. Results in *mode-seeking* approximations that concentrate on one mode. Variational inference minimises this.

Example — Mode-Seeking Behaviour of Reverse KL

p is a bimodal Gaussian mixture: $p = 0.5\,\mathcal{N}(-2, 1) + 0.5\,\mathcal{N}(+2, 1)$. Fitting $q = \mathcal{N}(\mu, \sigma^2)$ by minimising $D_{\mathrm{KL}}(q \,\|\, p)$:
The optimum is $\mu \approx \pm 2$ (either mode), $\sigma^2 \approx 1$ — q captures one mode and ignores the other. By contrast, minimising $D_{\mathrm{KL}}(p \,\|\, q)$ (MLE) gives $\mu = 0$, $\sigma^2 \approx 5$ — a wide Gaussian covering both modes but fitting neither well.
This mode-seeking property of reverse KL is the main source of variance underestimation in mean-field VI: the approximate posterior is narrower than the true posterior.

f-divergences. For a convex function $f : \mathbb{R}_+ \to \mathbb{R}$ with $f(1) = 0$, the f-divergence is:

$$D_f(P\|Q) = \int q(\mathbf{x})\, f\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right) d\mathbf{x}$$

Special cases: KL divergence ($f(t) = t\log t$), reverse KL ($f(t) = -\log t$), total variation ($f(t) = \frac{1}{2}|t-1|$), Hellinger ($f(t) = (\sqrt{t}-1)^2$), χ^2 divergence ($f(t) = (t-1)^2$). All f-divergences satisfy non-negativity and $D_f(P\|P) = 0$. The f-GAN framework (Nowozin et al., 2016) trains GANs to minimise any f-divergence using a variational lower bound: $D_f(P\|Q) = \sup_{T:\mathcal{X}\to\text{dom}_{f^*}} \mathbb{E}_P[T(\mathbf{x})] - \mathbb{E}_Q[f^*(T(\mathbf{x}))]$ where f^* is the Fenchel conjugate.

Total variation distance. $\mathrm{TV}(P, Q) = \frac{1}{2}\int |p - q|\, d\mu = \sup_A |P(A) - Q(A)|$. It satisfies the data processing inequality: $\mathrm{TV}(P_X, Q_X) \leq \mathrm{TV}(P, Q)$ for any mapping $X : \Omega \to \mathcal{X}$. Unlike KL, TV is symmetric and bounded: $\mathrm{TV} \in [0, 1]$. The Pinsker inequality connects TV to KL: $\mathrm{TV}(P, Q)^2 \leq \frac{1}{2} D_{\mathrm{KL}}(P\|Q)$, giving a useful upper bound on TV when KL is small.

Rényi divergence. For $\alpha > 0, \alpha \neq 1$:

$$D_\alpha(P\|Q) = \frac{1}{\alpha - 1} \log \int p^\alpha q^{1-\alpha}\, d\mu$$

In the limit $\alpha \to 1$: $D_1(P\|Q) = D_{\mathrm{KL}}(P\|Q)$. Properties: $D_\alpha(P\|Q)$ is non-decreasing in α, so $D_0(P\|Q) \leq D_{1/2}(P\|Q) \leq D_1(P\|Q) \leq D_2(P\|Q) \leq D_\infty(P\|Q)$. Rényi divergences appear in differential privacy (the Rényi DP framework) and PAC-Bayes bounds.

13.4 Conditional Probability and Bayes' Theorem

Conditional probability. For continuous random variables with joint density $p(\mathbf{x}, \mathbf{y})$:

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})} = \frac{p(\mathbf{y} \mid \mathbf{x})\, p(\mathbf{x})}{p(\mathbf{y})}$$

where $p(\mathbf{y}) = \int p(\mathbf{x}, \mathbf{y})\, d\mathbf{x}$ is the marginal. The formula requires $p(\mathbf{y}) > 0$.

Bayes' rule (continuous form)

$$p(\square \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \square)\, p(\square)}{p(\mathcal{D})}$$

where:

- $p(\square)$: *prior* — belief about parameters before observing data.
- $p(\mathcal{D} \mid \square)$: *likelihood* — probability of data given $\square$.
- $p(\square \mid \mathcal{D})$: *posterior* — updated belief after observing data.
- $p(\mathcal{D}) = \int p(\mathcal{D} \mid \square) p(\square)\, d\square$: *evidence* (marginal likelihood).

Intractability of the evidence. The marginalisation $p(\mathcal{D}) = \int p(\mathcal{D}|\square)p(\square)\, d\square$ is a high-dimensional integral (the parameter space of a neural network is $\mathbb{R}^{|\theta|}$ with $|\theta| \sim 10^6$–10^9). This intractability motivates all approximate inference methods (Part 16).

Conditional independence. $X \perp\!\!\!\perp Y \mid Z$ iff $p(X, Y|Z) = p(X|Z)p(Y|Z)$. In deep latent variable models, the latent variables $\mathbf{z}$ are chosen so that $\mathbf{x} \perp\!\!\!\perp \mathbf{x}' \mid \mathbf{z}$ (observations are conditionally independent given the latent code), making the likelihood factorisable over N examples.

Sequential Bayesian updating. When observations arrive sequentially $\mathcal{D}_1, \mathcal{D}_2, \ldots$, the posterior after step t becomes the prior at step $t+1$:

$$p(\square|\mathcal{D}_{1:t}) \propto p(\mathcal{D}_t|\square) \cdot p(\square|\mathcal{D}_{1:t-1})$$

This property is exploited in Bayesian continual learning: the network can be updated on new tasks without forgetting previous ones (catastrophic forgetting) provided an estimate of the posterior is maintained.

Example — MAP estimation as a limit case

The MAP (*Maximum A Posteriori*) estimator is:

$$\hat{\square}_{\text{MAP}} = \arg\max_{\square} \log p(\square|\mathcal{D}) = \arg\max_{\square} \left[\log p(\mathcal{D}|\square) + \log p(\square)\right]$$

With a Gaussian prior $p(\square) = \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$, the term $\log p(\square) = -\lambda\|\square\|^2/2 + \text{const}$ becomes L_2 regularisation (weight decay). Standard neural network training with weight decay is therefore MAP inference with a spherical Gaussian prior.

From posterior to prediction. The Bayesian predictive distribution is:

$$p(y_*|\mathbf{x}_*, \mathcal{D}) = \int p(y_*|\mathbf{x}_*, \square)\, p(\square|\mathcal{D})\, d\square$$

This is an expectation over the posterior: it incorporates model uncertainty. In practice it is approximated by (i) the MAP prediction (plug-in), (ii) deep ensembles (average over multiple runs), or (iii) variational BNNs (Bayes by Backprop).

Bayes' theorem for continuous distributions.

$$p(\square|\mathbf{x}) = \frac{p(\mathbf{x}|\square)\,p(\square)}{\int p(\mathbf{x}|\square')\,p(\square')\,d\square'} = \frac{p(\mathbf{x}|\square)\,p(\square)}{p(\mathbf{x})}$$

The denominator $p(\mathbf{x}) = \int p(\mathbf{x}|\square)p(\square)\,d\square$ is the *marginal likelihood* or *evidence*. It is typically intractable — the core difficulty motivating variational inference and MCMC.

Conjugate priors. A prior $p(\square|\square)$ is *conjugate* to the likelihood $p(\mathbf{x}|\square)$ if the posterior has the same functional form:

Likelihood	Conjugate prior	Posterior
Binomial(n,θ)	Beta(α,β)	Beta($\alpha + k$,$\beta + n - k$)
Poisson(λ)	Gamma(α,β)	Gamma($\alpha + \sum x_i$,$\beta + n$)
Gaussian(μ,σ^2)	Gaussian(μ_0,σ_0^2)	Closed form
Categorical($\mathbf{p}$)	Dirichlet($\square$)	Dirichlet($\square + \mathbf{n}$)

Conjugacy enables exact Bayesian updates in closed form — exploited in CAVI (§16.3) and the E-step of EM (§15.2) for models with conjugate structure.

Total probability and marginalisation in DGMs. For a latent variable model $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})\,p(\mathbf{z})\,d\mathbf{z}$$

The intractability of this marginal is why exact MLE is infeasible for VAEs and GMMs — leading to EM and ELBO-based methods.

Bayesian sequential updating. If observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are conditionally independent given θ, Bayes' theorem applies sequentially: $p(\theta|\mathbf{x}_{1:n}) \propto p(\mathbf{x}_n|\theta)\,p(\theta|\mathbf{x}_{1:n-1})$. Each new observation updates the posterior, which becomes the new prior. This online Bayesian updating is used in streaming ML (Bayesian online learning) and Kalman filtering. The posterior after n observations is:

$$p(\theta|\mathbf{x}_{1:n}) = \frac{p(\mathbf{x}_n|\theta)\cdots p(\mathbf{x}_1|\theta)\,p(\theta)}{p(\mathbf{x}_{1:n})} = \frac{p(\theta)\prod_{i=1}^{n} p(\mathbf{x}_i|\theta)}{p(\mathbf{x}_{1:n})}$$

Conjugate priors. A prior $p(\theta|\psi)$ is *conjugate* to the likelihood $p(\mathbf{x}|\theta)$ if the posterior $p(\theta|\mathbf{x}, \psi)$ is in the same family as the prior. Conjugate pairs preserve the prior family under Bayesian updating, enabling closed-form posteriors:

- Gaussian likelihood + Gaussian prior ⇒ Gaussian posterior (Bayesian linear regression).
- Binomial likelihood + Beta prior ⇒ Beta posterior ($p \sim \text{Beta}(\alpha + k, \beta + n - k)$ after k successes in n trials).
- Multinomial likelihood + Dirichlet prior ⇒ Dirichlet posterior (Bayesian categorical models, LDA).
- Poisson likelihood + Gamma prior ⇒ Gamma posterior (count data models).

For exponential family likelihoods, conjugate priors always exist and the posterior update is: $p(\theta|\mathbf{x}, \eta_0, \nu_0) = p(\theta|\eta_0 + T(\mathbf{x}), \nu_0 + 1)$, where $T(\mathbf{x})$ is the sufficient statistic and η_0, ν_0 are hyperparameters.

13.5 Information Theory Foundations

Definition: Shannon Entropy

For a discrete distribution p over K outcomes:

$$H(p) = -\sum_{k=1}^{K} p_k \log p_k \quad \text{(in nats if log base } e\text{; bits if base 2)}$$

Properties. $H(p) \geq 0$; $H(p) = 0$ iff p is deterministic; $H(\text{Uniform}_K) = \log K$ (maximum entropy).

Conditional entropy:

$$H(Y|X) = \mathbb{E}_{p(X)}[-\log p(Y|X)] = H(X,Y) - H(X)$$

Mutual information:

$$I(X;Y) = H(Y) - H(Y|X) = D_{\text{KL}}(p(X,Y) \,\|\, p(X)p(Y)) \geq 0$$

$I(X;Y) = 0$ iff X and Y are independent.

Example — Entropy and cross-entropy in classification

True distribution: $p = (0.7, 0.2, 0.1)$ over 3 classes. Model output: $\hat{p} = (0.6, 0.3, 0.1)$.
$H(p) = -(0.7\log 0.7 + 0.2\log 0.2 + 0.1\log 0.1) = 0.7(0.357) + 0.2(1.609) + 0.1(2.303) = 0.802$ nats.
Cross-entropy: $H(p,\hat{p}) = -(0.7\log 0.6 + 0.2\log 0.3 + 0.1\log 0.1) = 0.7(0.511) + 0.2(1.204) + 0.1(2.303) = 0.828$ nats.
$D_{\text{KL}}(p \,\|\, \hat{p}) = H(p,\hat{p}) - H(p) = 0.828 - 0.802 = 0.026$ nats.

Data processing inequality. If $X \to Y \to Z$ form a Markov chain, then $I(X;Z) \leq I(X;Y)$. Processing cannot increase information: any deterministic or stochastic function of Y contains no more information about X than Y itself. In deep learning, each layer maps the previous layer's output; by DPI, later representations can only lose information about the input.

Information bottleneck. The information bottleneck (Tishby et al., 1999) frames learning as compressing X into a representation T that retains maximum information about target Y:

$$\min_{p(t|x)} I(X;T) - \beta\, I(T;Y)$$

β controls the compression-accuracy tradeoff. The optimal representation is a sufficient statistic of X for predicting Y. Tishby & Schwartz-Ziv (2017) argued that SGD training proceeds through two phases: fitting (increasing $I(T;Y)$) then compression (decreasing $I(X;T)$) — though this interpretation remains debated.

Cross-entropy in practice. For a classifier with softmax output $\hat{p}$ trained on one-hot targets p:

$$H(p,\hat{p}) = H(p) + D_{\text{KL}}(p \,\|\, \hat{p}) = -\log \hat{p}_y$$

Since $H(p) = 0$ (one-hot), cross-entropy loss $=$ KL$(p\|\hat{p})$. Minimising cross-entropy is equivalent to minimising KL divergence from the label distribution to the model's output distribution.

Entropy chain rule. For joint distribution $p(\mathbf{x}_1, \dots, \mathbf{x}_n)$:

$$H(\mathbf{x}_1, \dots, \mathbf{x}_n) = \sum_{i=1}^{n} H(\mathbf{x}_i | \mathbf{x}_1, \dots, \mathbf{x}_{i-1})$$

This factorisation underlies autoregressive models: $\log p(\mathbf{x}) = \sum_i \log p(x_i | x_1, \dots, x_{i-1})$.

Data processing inequality (DPI). For a Markov chain $X \to Y \to Z$:

$$I(X; Z) \leq I(X; Y) \leq H(X)$$

Processing can only decrease or preserve mutual information. This has profound implications:

- *Representations:* a representation $Z = f(Y)$ retains at most $I(X; Y)$ bits about X. Dimensionality reduction always loses information.
- *Invariant features:* if Z is invariant to transformations of Y (e.g. rotations), then $I(X; Z) < I(X; Y)$ — the invariance trades off information for robustness.
- *Sufficient statistics:* $T(Y)$ is sufficient for θ iff $I(\theta; T(Y)) = I(\theta; Y)$ — no information is lost.

Information bottleneck (IB). Tishby et al. (2000) formulate representation learning as:

$$\min_{p(Z|X)} I(X; Z) - \beta I(Y; Z)$$

Find a compressed representation Z that maximises relevance to the target Y (maximise $I(Y; Z)$) while minimising complexity (minimise $I(X; Z)$). The Lagrange multiplier β trades off compression vs. task performance. Deep learning is conjectured to implement IB implicitly (Schwartz-Ziv & Tishby, 2017), though this remains debated.

Cross-entropy and log-loss. For a model q evaluated on data from p:

$$H(p, q) = -\sum_x p(x) \log q(x) = H(p) + D_{\mathrm{KL}}(p \,\|\, q) \geq H(p)$$

The excess cross-entropy over entropy is exactly the KL divergence. Minimising cross-entropy (log-loss) in supervised learning minimises KL from the data distribution p to the model q, equivalent to MLE when $H(p)$ is fixed.

Example — Perplexity in language modelling

Perplexity is $2^{H(p,q)}$ (in nats: $e^{H(p,q)}$). For a language model q on a test corpus $\mathcal{D}$ of N tokens:

$$\mathrm{PPL} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N} \log q(x_i | x_1, \dots, x_{i-1})\right)$$

Perplexity k means the model is as uncertain as a uniform distribution over k tokens at each step. GPT-4 achieves $\sim$ 3–5 PPL on standard English benchmarks; human-level is $\sim$ 15–30 (humans are uncertain even about their own continuations).

13.6 Sufficient Statistics and the Fisher Information

Fisher information. For a parametric model $p(x; \theta)$, the Fisher information measures how much information a single observation carries about θ:

$$\mathcal{J}(\theta) = \mathbb{E}_{p(x;\theta)}\left[\left(\frac{\partial \log p(x;\theta)}{\partial \theta}\right)^2\right] = -\mathbb{E}_{p(x;\theta)}\left[\frac{\partial^2 \log p(x;\theta)}{\partial \theta^2}\right]$$

Cramér-Rao bound. For any unbiased estimator $\hat{\theta}$ of θ:

$$\text{Var}(\hat{\theta}) \geq \frac{1}{n\,\mathcal{I}(\theta)}$$

This sets a lower bound on the variance of any unbiased estimator. The MLE achieves this bound asymptotically (efficiency).

Fisher information matrix. For a vector parameter $\square \in \mathbb{R}^d$:

$$F_{ij}(\square) = \mathbb{E}\left[\frac{\partial \log p}{\partial \theta_i}\frac{\partial \log p}{\partial \theta_j}\right]$$

The Fisher information matrix is the covariance of the score function. It appears in the natural gradient (Volume IV, Part 40) and the Laplace approximation to the posterior.

PART 14

Exponential Family and Graphical Models

The exponential family unifies all common distributions under a single framework. Graphical models encode conditional independence structure compactly, enabling tractable inference.

Fisher information matrix (FIM). For a vector parameter $\square \in \mathbb{R}^n$:

$$\mathbf{F}(\square) = \mathbb{E}_{p(\mathbf{x};\square)}[\nabla_\theta \log p(\mathbf{x};\square)\, \nabla_\theta \log p(\mathbf{x};\square)^\top] = -\mathbb{E}[\mathbf{H}_\theta \log p(\mathbf{x};\square)]$$

The FIM defines a Riemannian metric on the statistical manifold: $ds^2 = d\square^\top \mathbf{F}(\square)\, d\square$ measures the "statistical distance" between nearby distributions.

Natural gradient. Ordinary gradient descent moves in Euclidean $\square$-space. The natural gradient (Amari, 1998) moves in the space of distributions, using the FIM as a preconditioner:

$$\tilde{\nabla}_\theta \mathcal{L} = \mathbf{F}(\square)^{-1} \nabla_\theta \mathcal{L}$$

Natural gradient descent is invariant to reparameterisation of the model and converges faster than SGD near the optimum. K-FAC (Kronecker-Factored Approximate Curvature) approximates $\mathbf{F}^{-1}$ as a Kronecker product for efficiency.

Connection to Adam. Adam's denominator $\sqrt{\hat{v}_t}$ approximates the diagonal of $\mathbf{F}^{1/2}$: it estimates the RMS gradient (related to the square root of the Fisher diagonal). Adam can be viewed as a diagonal natural gradient preconditioner.

EXAMPLE — Fisher Information for Bernoulli Distribution

$p(x;\theta) = \theta^x(1-\theta)^{1-x}$, $x \in \{0,1\}$.
$\log p = x\log\theta + (1-x)\log(1-\theta)$.
$\frac{\partial \log p}{\partial \theta} = \frac{x}{\theta} - \frac{1-x}{1-\theta}$.
$\mathcal{I}(\theta) = \mathbb{E}\left[\left(\frac{X}{\theta} - \frac{1-X}{1-\theta}\right)^2\right] = \frac{\theta}{\theta^2} + \frac{1-\theta}{(1-\theta)^2} = \frac{1}{\theta(1-\theta)}$.
Cramér-Rao: $\mathrm{Var}(\hat{\theta}) \geq \theta(1-\theta)/n$. For $n = 100$, $\theta = 0.5$: Var ≥ 0.0025, so std ≥ 0.05. The MLE $\hat{\theta} = \bar{X}$ achieves this bound exactly.

Cramér-Rao lower bound (CRLB). For any unbiased estimator $\hat{\theta}(\mathbf{x})$ of θ:

$$\mathrm{Var}(\hat{\theta}) \geq \frac{1}{\mathcal{I}(\theta)}$$

The variance is bounded below by the reciprocal of the Fisher information. An estimator achieving this bound is *efficient*. The MLE achieves the CRLB asymptotically: $\sqrt{n}(\hat{\theta}_{\text{MLE}} - \theta) \xrightarrow{d} \mathcal{N}(0, \mathcal{I}(\theta)^{-1})$.

Fisher information matrix (FIM) in neural networks. For a model $p_\theta(\mathbf{x})$ with parameter vector $\theta \in \mathbb{R}^p$, the FIM is:

$$\mathbf{F}(\theta) = \mathbb{E}_{p_\theta(\mathbf{x})}[\nabla_\theta \log p_\theta(\mathbf{x})\, (\nabla_\theta \log p_\theta(\mathbf{x}))^\top] \in \mathbb{R}^{p\times p}$$

The FIM defines the local geometry of the parameter space in the sense of the Fisher metric. Natural gradient descent uses $\mathbf{F}^{-1}$ as a preconditioner:

$$\theta \leftarrow \theta - \eta\, \mathbf{F}(\theta)^{-1} \nabla_\theta \mathcal{L}(\theta)$$

The natural gradient is steepest descent in the space of distributions (under KL divergence), not in parameter space. For neural networks, the FIM is $p \times p$ (e.g. $10^9 \times 10^9$) — computing and inverting it exactly is infeasible; approximations (K-FAC, Kronecker-factored approximate curvature) are used.

Sufficiency and the Fisher-Neyman factorisation theorem. $T(\mathbf{x})$ is a sufficient statistic for θ iff the likelihood factorises:

$$p(\mathbf{x}|\theta) = h(\mathbf{x}) \cdot g(T(\mathbf{x}), \theta)$$

Equivalently, T is sufficient iff the posterior $p(\theta|\mathbf{x}) = p(\theta|T(\mathbf{x}))$: all information about θ in $\mathbf{x}$ is captured by T. For exponential family $p(\mathbf{x}|\theta) = h(\mathbf{x}) \exp(\eta(\theta)^\top T(\mathbf{x}) - A(\theta))$, the natural sufficient statistic is $T(\mathbf{x})$ (the canonical statistic). In neural networks, learned representations can be viewed as (approximately) sufficient statistics for the label y.

14.1 The Exponential Family

Definition: Exponential Family (canonical form)

A distribution belongs to the *exponential family* if its density/mass can be written as:

$$p(\mathbf{x}; \boldsymbol{\eta}) = h(\mathbf{x}) \exp(\boldsymbol{\eta}^\top \mathbf{T}(\mathbf{x}) - A(\boldsymbol{\eta}))$$

where:

- $\boldsymbol{\eta}$: natural (canonical) parameters.
- $\mathbf{T}(\mathbf{x})$: sufficient statistics.
- $h(\mathbf{x})$: base measure.
- $A(\boldsymbol{\eta}) = \log \int h(\mathbf{x}) \exp(\boldsymbol{\eta}^\top \mathbf{T}(\mathbf{x}))\, d\mathbf{x}$: log-partition function (cumulant function).

Properties of the log-partition function.

$$\nabla_{\boldsymbol{\eta}} A = \mathbb{E}_{\boldsymbol{\eta}}[\mathbf{T}(\mathbf{X})], \qquad \nabla^2_{\boldsymbol{\eta}} A = \mathrm{Cov}_{\boldsymbol{\eta}}[\mathbf{T}(\mathbf{X})]$$

The Hessian of A is the covariance matrix of the sufficient statistics, which is positive semi-definite. Hence A is *convex* and the mapping $\boldsymbol{\eta} \mapsto \mathbb{E}[\mathbf{T}]$ is bijective (under regularity conditions). This convexity is crucial for the convergence of CAVI.

MLE for the exponential family. Maximum likelihood reduces to moment matching:

$$\hat{\boldsymbol{\eta}}_{\text{MLE}} : \quad \mathbb{E}_{\hat{\boldsymbol{\eta}}}[\mathbf{T}(\mathbf{X})] = \frac{1}{n} \sum_{i=1}^{n} \mathbf{T}(\mathbf{x}_i)$$

This is the unique first-order condition and, by the convexity of $-\ell(\boldsymbol{\eta})$, is also globally optimal.

Example — Canonical examples

- **Bernoulli**: $p(x; \eta) = (1 + e^{-\eta})^{-x} \cdot (1 + e^{\eta})^{-(1-x)}$, $\eta = \log \frac{\mu}{1-\mu}$ (logit), $T(x) = x$, $A(\eta) = \log(1 + e^{\eta})$.
- **Gaussian**: $\boldsymbol{\eta} = (\mu/\sigma^2,\ -1/(2\sigma^2))^\top$, $\mathbf{T}(x) = (x, x^2)^\top$, $A = \mu^2/(2\sigma^2) + \frac{1}{2}\log(2\pi\sigma^2)$.
- **Poisson**: $\eta = \log \lambda$, $T(x) = x$, $h(x) = 1/x!$, $A(\eta) = e^{\eta}$.
- **Categorical** (K classes): $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_{K-1})^\top$, $T_k(x) = \mathbf{1}[x = k]$, $A = \log(1 + \sum_{k=1}^{K-1} e^{\eta_k})$.

- **Multivariate Gaussian**: $\boldsymbol{\eta} = (\square^{-1}\boldsymbol{\mu},\ -\frac{1}{2}\square^{-1})$, vectorised form.

Relevance for neural networks. The most common activation functions and loss functions are derived directly from the exponential family: the sigmoid is the mean-response function for Bernoulli, softmax for categorical, the identity for Gaussian. Cross-entropy loss is the negative log-likelihood of the corresponding exponential family distribution.

Conjugate prior. A prior $p(\boldsymbol{\eta}) \propto \exp(\boldsymbol{\chi}^\top \boldsymbol{\eta} - \nu A(\boldsymbol{\eta}))$ is conjugate to the exponential family likelihood: the posterior belongs to the same family with updated parameters $\boldsymbol{\chi}' = \boldsymbol{\chi} + \sum_i \mathbf{T}(\mathbf{x}_i)$, $\nu' = \nu + n$. This enables exact inference in many graphical models (Part 14) and motivates Gaussian priors for continuous latent variables.

Definition: Exponential Family

A family of distributions parameterised by $\square \in \mathbb{R}^k$ (natural parameters):

$$p(\mathbf{x} \mid \square) = h(\mathbf{x}) \exp(\square^\top T(\mathbf{x}) - A(\square))$$

where:

- $T(\mathbf{x})$: *sufficient statistic* — the data only matters through $T(\mathbf{x})$.
- $h(\mathbf{x})$: base measure.
- $A(\square) = \log \int h(\mathbf{x}) \exp(\square^\top T(\mathbf{x}))\, d\mathbf{x}$: *log-partition function* (normaliser).

Key property. The log-partition function generates cumulants:

$$\mathbb{E}[T(\mathbf{x})] = \nabla_\eta A(\square), \qquad \mathrm{Cov}[T(\mathbf{x})] = \nabla_\eta^2 A(\square)$$

Examples:

Distribution	$T(x)$	$\square$	$A(\square)$
$\mathcal{N}(\mu, \sigma^2)$	(x, x^2)	$(\mu/\sigma^2, -1/2\sigma^2)$	$-\eta_1^2/4\eta_2 - \frac{1}{2}\log(-2\eta_2)$
Bernoulli(p)	x	$\log \frac{p}{1-p}$	$\log(1 + e^\eta)$
Poisson(λ)	x	$\log \lambda$	e^η
Categorical$(\mathbf{p})$	$(1_{x=1}, \ldots)$	$(\log p_1, \ldots)$	$\log \sum_k e^{\eta_k}$

14.2 Directed Graphical Models (Bayesian Networks)

Bayesian Network

A *directed graphical model* (DAG) is a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ over variables $\{X_v\}_{v \in \mathcal{V}}$. The joint distribution factorises as:

$$p(\mathbf{x}) = \prod_{v \in \mathcal{V}} p\big(x_v \mid x_{\mathrm{pa}(v)}\big)$$

where $\mathrm{pa}(v)$ denotes the parents of v in the graph.

d-Separation. Two sets A, B are conditionally independent given C (i.e. $A \perp\!\!\!\perp B \mid C$) iff every undirected path between A and B is *blocked* by C. A path is blocked by C if:

1. It contains a *chain* $X \to Z \to Y$ or *fork* $X \leftarrow Z \to Y$ with $Z \in C$; or

2. It contains a *collider* $X \to Z \leftarrow Y$ with $Z \notin C$ and no descendant of Z in C.

d-Separation guarantees conditional independence in the distribution induced by the DAG.

Latent variable models as DAGs. The VAE has the structure:

$$p(\mathbf{z}) \;\longrightarrow\; \mathbf{z} \;\longrightarrow\; \mathbf{x}$$

Gaussian mixture models: $p(\pi) \to c \to \mathbf{x} \leftarrow \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c$. Sequence models (HMM): $z_1 \to z_2 \to \cdots \to z_T$ with $z_t \to x_t$.

Example — Factorisation of a Gaussian mixture model

With K components and N observed points:

$$\begin{aligned} \boldsymbol{\pi} &\sim \text{Dir}(\boldsymbol{\alpha}) \\ c_i|\boldsymbol{\pi} &\sim \text{Cat}(\boldsymbol{\pi}), \quad i = 1, \dots, N \\ \mathbf{x}_i|c_i, \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\} &\sim \mathcal{N}(\boldsymbol{\mu}_{c_i}, \boldsymbol{\Sigma}_{c_i}) \end{aligned}$$

The complete likelihood (with latent variables c_i) is tractable; the marginal $p(\mathbf{x}_i) = \sum_k \pi_k \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the mixture.

Exact vs. approximate inference.

- **Variable elimination**: exact, exponential complexity in treewidth.
- **Belief propagation**: exact on trees, approximate on loopy graphs (loopy BP).
- **Variational inference (VI)**: approximates the posterior with a tractable family q by maximising the ELBO (Part 16).
- **MCMC**: samples from the exact posterior in the limit of infinite runs; expensive for neural networks.

Plate notation. The N i.i.d. data points are represented with a *plate*: a rectangle around the replicated variables labelled N. The VAE with N points has a plate containing $(\mathbf{x}_i, \mathbf{z}_i)$ and parameters θ, ϕ outside the plate (shared).

A directed graphical model (Bayesian network) over variables $\{X_1, \dots, X_n\}$ is a DAG G together with conditional distributions $p(X_i \mid \text{Pa}(X_i))$, where $\text{Pa}(X_i)$ are the parents of node i in G. The joint factorises as:

$$p(X_1, \dots, X_n) = \prod_{i=1}^{n} p(X_i \mid \text{Pa}(X_i))$$

d-separation. Two nodes X and Y are *d-separated* by a set Z if all paths between them are blocked. d-separation implies conditional independence: $X \perp Y \mid Z$.

Latent variable model. Introduce unobserved variables $\mathbf{z}$ (latent variables) to explain structure in observed data $\mathbf{x}$:

$$p(\mathbf{x}) = \int p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}$$

The marginalisation over $\mathbf{z}$ is usually intractable — this integral is the core challenge of latent variable models.

D-separation and conditional independence. A path between nodes A and B is *blocked* by a set C if:

- The path contains a *chain* $A \to m \to B$ or *fork* $A \leftarrow m \to B$ and $m \in C$, or
- The path contains a *collider* $A \to m \leftarrow B$ and neither m nor any of its descendants is in C.

A and B are *d-separated* by C if all paths between them are blocked. By the d-separation theorem, if $A \perp_G B \mid C$ (d-separated in the graph), then $A \perp_P B \mid C$ (conditionally independent under any distribution P that factorises over G).

Plate notation. Repeated variables are represented with *plates* — rectangles enclosing replicated subgraphs with an index label. For a Bayesian linear regression model with N observations:

- Outer plate: $n = 1, \dots, N$ (data points). Inside: $\mathbf{x}_n \to y_n \leftarrow \mathbf{w}$.
- $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$ outside the plate (shared across observations).
- $y_n \sim \mathcal{N}(\mathbf{w}^\top \mathbf{x}_n, \sigma^2)$ inside the plate.

Plate notation makes the joint factorisation explicit: $p(\mathbf{w}, \{y_n\}) = p(\mathbf{w}) \prod_{n=1}^{N} p(y_n | \mathbf{x}_n, \mathbf{w})$.

Explaining away. Colliders create a counter-intuitive conditional dependence: if $A \to C \leftarrow B$ and A, B are marginally independent, conditioning on C makes A and B conditionally dependent. Example: A = "rained", B = "sprinkler on", C = "grass wet". Knowing the grass is wet (observing C), learning it rained makes the sprinkler being on less likely — the causes "explain away" each other.

14.3 Undirected Graphical Models (Markov Random Fields)

An undirected graphical model (MRF) over variables $\{X_1, \dots, X_n\}$ is an undirected graph G with potential functions $\psi_C(\mathbf{x}_C)$ defined over cliques $C \subseteq V$:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{C} \psi_C(\mathbf{x}_C), \qquad Z = \sum_{\mathbf{x}} \prod_{C} \psi_C(\mathbf{x}_C)$$

The normalisation constant Z (partition function) is generally intractable. This is the core challenge of undirected models. **Restricted Boltzmann Machine (RBM).** A bipartite MRF with visible units $\mathbf{v} \in \{0,1\}^d$ and hidden units $\mathbf{h} \in \{0,1\}^p$:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp(\mathbf{v}^\top \mathbf{W} \mathbf{h} + \mathbf{b}^\top \mathbf{v} + \mathbf{c}^\top \mathbf{h})$$

The bipartite structure means $p(\mathbf{h}|\mathbf{v}) = \prod_j p(h_j|\mathbf{v})$ and $p(\mathbf{v}|\mathbf{h}) = \prod_i p(v_i|\mathbf{h})$ (conditionally independent given the other layer). Contrastive divergence (§22.3) trains RBMs by approximating the negative phase with Gibbs sampling.

Comparison: directed vs. undirected models.

Property	Bayesian network (directed)	MRF (undirected)
Normalisation	Always normalised	Requires partition Z
Conditional indep.	D-separation	Graph separation
Inference	Forward-backward	Belief propagation
Learning	MLE (tractable)	Requires MCMC/NCE
Examples	HMM, Naive Bayes	RBM, CRF, Ising model

Clique factorisation. The Hammersley-Clifford theorem states that a positive distribution $p(\mathbf{x}) > 0$ is Markov with respect to a graph G iff it factorises over the cliques of G:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C)$$

where $\mathcal{C}$ is the set of maximal cliques and $\psi_C : \mathcal{X}_C \to \mathbb{R}_{>0}$ are *potential functions* (not necessarily probabilities). The partition function $Z = \sum_{\mathbf{x}} \prod_C \psi_C(\mathbf{x}_C)$ normalises the distribution.

Gibbs distribution. Setting $\psi_C(\mathbf{x}_C) = \exp(-E_C(\mathbf{x}_C))$ for clique energies E_C:

$$p(\mathbf{x}) = \frac{1}{Z} \exp\left(-\sum_C E_C(\mathbf{x}_C) \right) = \frac{1}{Z} \exp(-E(\mathbf{x}))$$

This is the Gibbs (or Boltzmann) distribution. Energy-based models (EBMs, §22.1) are MRFs with learned clique energies. The global energy $E(\mathbf{x}) = \sum_C E_C(\mathbf{x}_C)$ can be any function; the partition function Z handles normalisation.

Conditional independence in MRFs. Three key Markov properties:

- *Pairwise Markov:* $X_i \perp X_j \mid \mathbf{X}_{V \setminus \{i,j\}}$ for non-adjacent i, j.
- *Local Markov:* $X_i \perp \mathbf{X}_{V \setminus \mathrm{ne}(i) \setminus \{i\}} \mid \mathbf{X}_{\mathrm{ne}(i)}$ (a node is independent of all non-neighbours given its neighbours).
- *Global Markov:* $\mathbf{X}_A \perp \mathbf{X}_B \mid \mathbf{X}_S$ whenever S *separates* A from B in G.

These three are equivalent for positive distributions. Separation (d-separation analogue for undirected graphs) is simply: all paths from A to B pass through S.

Belief propagation on MRFs. Exact marginal inference via variable elimination (or equivalent sum-product BP) runs in time exponential in the tree-width. For tree-structured MRFs, BP is exact and runs in $O(N|\mathcal{X}|^2)$. For loopy graphs, loopy BP passes messages iteratively until convergence — it is not guaranteed to converge but works well empirically (used in JPEG2000 decoding, stereo vision, turbo codes).

Example — Ising model as a pairwise MRF

The Ising model on a grid graph $G = (V, E)$ with binary spins $x_i \in \{-1, +1\}$:

$$p(\mathbf{x}) = \frac{1}{Z} \exp\left(J \sum_{(i,j) \in E} x_i x_j + h \sum_i x_i \right)$$

is a pairwise MRF with edge potentials $\psi_{ij}(x_i, x_j) = \exp(J x_i x_j)$ and node potentials $\psi_i(x_i) = \exp(h x_i)$. Computing Z is #P-hard for general graphs, but polynomial for planar graphs (Kasteleyn's formula). The Ising model exhibits a second-order phase transition at the critical temperature $T_c = 2J / \ln(1 + \sqrt{2})$ (2D square lattice).

14.4 Hierarchical Bayesian Models

In a hierarchical model, hyperparameters □ govern the prior over parameters □, which in turn govern the observations $\mathbf{x}$:

$$p(\mathbf{x}, \square, \square) = p(\mathbf{x}|\square)\, p(\square|\square)\, p(\square)$$

Empirical Bayes. Integrate out □ and maximise over □:

$$\hat{□} = \arg\max_{□} \log p(\mathbf{x}|□) = \arg\max_{□} \log \int p(\mathbf{x}|□)p(□|□)\, d□$$

This marginalises over parameters (Bayesian) but estimates hyperparameters by MLE (frequentist) — a compromise that often works well in practice.

Connection to deep learning. Neural networks with L_2 regularisation correspond to a hierarchical model where parameters have a Gaussian prior $□ \sim \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$. Weight decay coefficient λ plays the role of the hyperparameter □.

PART 15

The EM Algorithm

Expectation-Maximisation alternates between inferring latent variable distributions (E-step) and maximising parameters (M-step). We derive both steps and prove the monotone likelihood increase.

Example: Bayesian neural network. A BNN treats the weights $\mathbf{W}$ as random variables with prior $p(\mathbf{W})$ (e.g., $\mathcal{N}(\mathbf{0}, \sigma_0^2\mathbf{I})$). The posterior:

$$p(\mathbf{W}|\mathcal{D}) \propto p(\mathcal{D}|\mathbf{W})\, p(\mathbf{W})$$

is intractable for deep networks (no conjugacy). Variational inference (Part 16) approximates it with $q_\phi(\mathbf{W})$.

Hierarchical priors for regularisation. A two-level hierarchy:

$$\sigma_l^2 \sim \text{InvGamma}(a, b), \quad w_{lij} \sim \mathcal{N}(0, \sigma_l^2)$$

The layer-wise variance σ_l^2 is a hyperparameter learned from data. This is an *automatic relevance determination* (ARD) prior: layers with little useful information shrink to $\sigma_l^2 \to 0$, effectively pruning the network.

Practical implications. Hierarchical models naturally handle:

- *Partial pooling:* observations from different groups share a common hyperprior, enabling information transfer between groups. Example: learning language models across multiple languages with shared subword vocabulary statistics.
- *Uncertainty quantification:* posterior predictive $p(y^*|\mathbf{x}^*, \mathcal{D}) = \int p(y^*|\mathbf{x}^*, \square)\, p(\square|\mathcal{D})\, d\square$ marginalises over parameter uncertainty, giving calibrated predictions.
- *Meta-learning:* MAML (Model-Agnostic Meta-Learning) can be interpreted as empirical Bayes: the shared initialisation is the hyperparameter optimised across tasks.

Plate notation. Hierarchical models are compactly represented with *plates* — boxes indicating replication. A two-level model for N observations:

$$\alpha \to \theta_n \to x_{n,m} \quad (n = 1, \ldots, N;\ m = 1, \ldots, M)$$

where α is a hyperprior, θ_n are per-group parameters, and $x_{n,m}$ are observations within group n. The joint:

$$p(\mathbf{x}, \square, \alpha) = p(\alpha) \prod_{n=1}^{N} p(\theta_n|\alpha) \prod_{m=1}^{M} p(x_{n,m}|\theta_n)$$

Empirical Bayes (Type-II MLE). Rather than placing a full prior on α and integrating, empirical Bayes estimates $\hat{\alpha}$ by maximising the marginal likelihood:

$$\hat{\alpha} = \arg\max_{\alpha} \prod_{n=1}^{N} \int p(x_n|\theta_n) p(\theta_n|\alpha)\, d\theta_n$$

This is the *evidence* (marginal likelihood) maximised over α. Empirical Bayes avoids specifying a hyperprior at the cost of ignoring uncertainty in α — it typically underestimates posterior variance.

Partial pooling. Hierarchical models implement *partial pooling* between:

- *No pooling:* estimate θ_n independently for each group. High variance for small groups.
- *Complete pooling:* estimate one θ for all groups. High bias if groups differ.
- *Partial pooling (hierarchical):* $\theta_n \sim p(\theta|\alpha)$ shrinks group estimates toward the global mean by an amount depending on group size and variability.

The shrinkage is adaptive: small groups (few data) are shrunk more toward the global mean; large groups resist shrinkage. Mathematically, for Gaussian hierarchical models this is equivalent to James-Stein shrinkage.

Deep generative models as hierarchical Bayes. VAEs implement a two-level hierarchical model:

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$$

The prior $p(\mathbf{z})$ is the top-level prior; the decoder $p_\theta(\mathbf{x}|\mathbf{z})$ is the likelihood. Hierarchical VAEs (NVAE, VDVAE) add multiple stochastic layers $\mathbf{z}_1, \ldots, \mathbf{z}_L$ with conditional priors $p_\theta(\mathbf{z}_\ell|\mathbf{z}_{<\ell})$, enabling richer latent structure and sharper generated images.

15.1 The Log-Likelihood Decomposition

Objective. In latent variable models we want to maximise the marginal log-likelihood:

$$\ell(\theta) = \log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}, \mathbf{z})\, d\mathbf{z}$$

This integral is intractable. The key decomposition circumvents it by introducing an auxiliary distribution $q(\mathbf{z})$:

ELBO + Gap Decomposition

For any $q(\mathbf{z})$ with $\operatorname{supp}(q) \supseteq \operatorname{supp}(p(\cdot|\mathbf{x}))$:

$$\log p_\theta(\mathbf{x}) = \underbrace{\mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})}\right]}_{\mathcal{L}_{\text{ELBO}}(q,\theta)} + \underbrace{D_{\text{KL}}(q(\mathbf{z}) \,\|\, p_\theta(\mathbf{z}|\mathbf{x}))}_{\geq 0}$$

Proof of the decomposition.

$$\begin{aligned}
\log p_\theta(\mathbf{x}) &= \mathbb{E}_{q(\mathbf{z})}[\log p_\theta(\mathbf{x})] \\
&= \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right] \\
&= \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})}\right] + \mathbb{E}_q\left[\log \frac{q(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right] \\
&= \mathcal{L}_{\text{ELBO}}(q, \theta) + D_{\text{KL}}(q \,\|\, p_\theta(\cdot|\mathbf{x}))
\end{aligned}$$

Corollary: ELBO as a lower bound. Since $D_{\text{KL}}(\geq \| 0)$:

$$\log p_\theta(\mathbf{x}) \geq \mathcal{L}_{\text{ELBO}}(q, \theta)$$

with equality iff $q(\mathbf{z}) = p_\theta(\mathbf{z}|\mathbf{x})$ a.e. Maximising the ELBO is equivalent to reducing the KL gap.

ELBO rewritten. Expanding the joint:

$$\mathcal{L}_{\text{ELBO}}(q, \theta) = \mathbb{E}_q[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{\text{KL}}(q(\mathbf{z}) \,\|\, p(\mathbf{z}))$$

The first term is the *reconstruction accuracy*; the second is the *regularisation* term that pushes q toward the prior. This tradeoff is the heart of the VAE (β-VAE weights the second term by β).

Example — Geometry of the ELBO

In the space of distributions q, the ELBO is a concave function of q (for fixed θ). Maximising over $\mathcal{Q}$ yields the q^* that minimises the KL to the true posterior. If $\mathcal{Q}$ is the mean-field family we obtain CAVI; if it is the parametric family $\{q_\phi\}$ we obtain the VAE.

Connection to the MDL principle. The ELBO can be interpreted in the *Minimum Description Length* framework: $-\mathbb{E}_q[\log p_\theta(\mathbf{x}|\mathbf{z})]$ is the description cost of the data given the latent code, and $D_{\text{KL}}(q \,\|\, p)$ is the description cost of the code itself. Maximising the ELBO minimises the total description length. Given observed data $\mathbf{x}$ and latent variables $\mathbf{z}$, with parameters $\square$, we want to maximise:

$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x} \,|\, \mathbf{z})\, p_\theta(\mathbf{z})\, d\mathbf{z}$$

This is intractable. Introduce any distribution $q(\mathbf{z})$ over latent variables and write:

$$\log p_\theta(\mathbf{x}) = \log \int q(\mathbf{z}) \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})}\, d\mathbf{z}$$

Apply Jensen's inequality ($\log$ is concave, so $\log \mathbb{E}[\cdot] \geq \mathbb{E}[\log(\cdot)]$):

$$\geq \int q(\mathbf{z}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})}\, d\mathbf{z} \triangleq \mathcal{L}(q, \square)$$

The gap between $\log p_\theta(\mathbf{x})$ and the lower bound $\mathcal{L}$ is exactly $D_{\text{KL}}(q(\mathbf{z}) \,\|\, p_\theta(\mathbf{z} \,|\, \mathbf{x}))$:

$$\log p_\theta(\mathbf{x}) = \mathcal{L}(q, \square) + D_{\text{KL}}(q(\mathbf{z}) \,\|\, p_\theta(\mathbf{z} \,|\, \mathbf{x}))$$

Derivation of the gap:

$$\begin{aligned} D_{\text{KL}}(q \,\|\, p_\theta(\cdot \,|\, \mathbf{x})) &= \int q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p_\theta(\mathbf{z} \,|\, \mathbf{x})}\, d\mathbf{z} \\ &= \int q(\mathbf{z}) \left[\log q(\mathbf{z}) - \log p_\theta(\mathbf{z} \,|\, \mathbf{x})\right] d\mathbf{z} \\ &= -\int q(\mathbf{z}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})}\, d\mathbf{z} + \log p_\theta(\mathbf{x}) \\ &= -\mathcal{L}(q, \square) + \log p_\theta(\mathbf{x}) \end{aligned}$$

15.2 The E-Step and M-Step

Algorithm: Expectation-Maximisation

Initialise $\square^{(0)}$. Iterate until convergence:
E-step: Fix $\square^{(t)}$. Set $q(\mathbf{z}) = p_{\theta^{(t)}}(\mathbf{z} \,|\, \mathbf{x})$.

This makes $D_{\mathrm{KL}}(q \,\|\, p_\theta(\cdot \,|\, \mathbf{x})) = 0$, so $\mathcal{L}(q, \square^{(t)}) = \log p_{\theta^{(t)}}(\mathbf{x})$.
M-step: Fix q. Maximise $\mathcal{L}(q, \square)$ over $\square$:

$$\square^{(t+1)} = \arg\max_{\square} \mathbb{E}_{q(\mathbf{z})}[\log p_\theta(\mathbf{x}, \mathbf{z})]$$

(the $-\int q \log q \, d\mathbf{z}$ entropy term does not depend on $\square$)

Monotone likelihood increase. After one E+M cycle:

$$\log p_{\theta^{(t+1)}}(\mathbf{x}) \geq \mathcal{L}(q^{(t+1)}, \square^{(t+1)}) \geq \mathcal{L}(q^{(t+1)}, \square^{(t)}) = \log p_{\theta^{(t)}}(\mathbf{x})$$

The first $\geq$ follows from $D_{\mathrm{KL}}(\geq \,\|\, 0)$; the second $\geq$ from the M-step maximisation; the equality from the E-step setting KL to zero.

Example — EM for a 1D Gaussian mixture ($K = 2$)

Model: $p(x) = \pi_1 \mathcal{N}(x; \mu_1, \sigma^2) + \pi_2 \mathcal{N}(x; \mu_2, \sigma^2)$, $\pi_1 + \pi_2 = 1$.
Latent variable $z \in \{1, 2\}$ indicates which component generated x.
E-step: Compute responsibilities $r_{ik} = p(z_i = k \,|\, x_i, \square^{(t)})$:

$$r_{ik} = \frac{\pi_k^{(t)} \mathcal{N}(x_i; \mu_k^{(t)}, \sigma^2)}{\sum_{j=1}^{2} \pi_j^{(t)} \mathcal{N}(x_i; \mu_j^{(t)}, \sigma^2)}$$

M-step: Re-estimate parameters:

$$\mu_k^{(t+1)} = \frac{\sum_i r_{ik} \, x_i}{\sum_i r_{ik}}, \qquad \pi_k^{(t+1)} = \frac{1}{N} \sum_i r_{ik}$$

Numerical step. Data: $x_1 = 1.0$, $x_2 = 2.0$, $x_3 = 5.0$, $x_4 = 6.0$. Init: $\mu_1 = 0$, $\mu_2 = 3$, $\pi_1 = \pi_2 = 0.5$, $\sigma^2 = 1$.
$r_{11} \propto \mathcal{N}(1; 0, 1) = 0.2420$, $\mathcal{N}(1; 3, 1) = 0.0540$: $r_{11} = 0.2420/0.2960 = 0.8176$
$r_{13} \propto \mathcal{N}(5; 0, 1) \approx 0$, $\mathcal{N}(5; 3, 1) = 0.0540$: $r_{13} \approx 0.0$ (component 2 gets this point)
After M-step: $\mu_1^{(1)} \approx 1.5$, $\mu_2^{(1)} \approx 5.5$ — the two clusters separate.

EM as coordinate ascent on the ELBO. Define the ELBO (Free Energy):

$$\mathcal{F}(q, \theta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}, \mathbf{z}; \theta)] + H(q)$$

EM alternates:

- *E-step:* $q^{(t+1)} = \arg\max_q \mathcal{F}(q, \theta^{(t)}) = p(\mathbf{z}|\mathbf{x}; \theta^{(t)})$. Setting q to the exact posterior makes $\mathcal{F} = \log p(\mathbf{x}; \theta^{(t)})$ (the KL gap disappears).
- *M-step:* $\theta^{(t+1)} = \arg\max_\theta \mathcal{F}(q^{(t+1)}, \theta) = \arg\max_\theta Q(\theta|\theta^{(t)})$.

Since each step increases $\mathcal{F}$ and $\mathcal{F} \leq \log p(\mathbf{x}; \theta)$, the log-likelihood is non-decreasing: $\log p(\mathbf{x}; \theta^{(t+1)}) \geq \log p(\mathbf{x}; \theta^{(t)})$.

Generalised EM. When the E-step is intractable (posterior has no closed form), we can:

- *Variational EM (VEM):* replace exact posterior with a variational approximation $q_\phi \approx p(\mathbf{z}|\mathbf{x}; \theta)$, optimising ϕ by CAVI or BBVI in the E-step.
- *Monte Carlo EM (MCEM):* approximate $Q(\theta|\theta^{(t)})$ by sampling from the posterior via MCMC.

The VAE can be seen as amortised VEM: the encoder $q_\phi(\mathbf{z}|\mathbf{x})$ is the amortised E-step, the decoder $p_\theta(\mathbf{x}|\mathbf{z})$ is the M-step.

Generalised EM (GEM). Standard EM requires maximising $Q(\theta;\theta^{(t)})$ exactly. GEM only requires an increase: $Q(\theta^{(t+1)};\theta^{(t)}) \geq Q(\theta^{(t)};\theta^{(t)})$. Since the M-step increase implies a log-likelihood increase, GEM also converges. Variants:

- *Expectation-Conditional Maximisation (ECM):* replace the M-step by multiple conditional maximisation steps, each increasing Q in a subset of parameters. Useful when the full joint maximisation is intractable.
- *Monte Carlo EM (MCEM):* when the E-step integral $\int q(\mathbf{z}) \log p(\mathbf{x},\mathbf{z}|\theta)\, d\mathbf{z}$ is intractable, approximate by sampling $\mathbf{z}^{(1)},\dots,\mathbf{z}^{(S)} \sim p(\mathbf{z}|\mathbf{x},\theta^{(t)})$ via MCMC.
- *Variational EM (VEM):* replace the exact posterior in the E-step with a variational approximation $q_\phi(\mathbf{z})$ from a tractable family. The VAE is VEM with amortised inference.

Convergence theorem (Dempster, Laird, Rubin, 1977). Every EM iterate satisfies $\log p(\mathbf{x}|\theta^{(t+1)}) \geq \log p(\mathbf{x}|\theta^{(t)})$. If the log-likelihood is bounded above, the sequence $\{\log p(\mathbf{x}|\theta^{(t)})\}$ converges. Under regularity conditions, $\theta^{(t)} \to \theta^*$ where θ^* is a stationary point (but not necessarily the global maximum) of $\log p(\mathbf{x}|\theta)$.

Proof sketch. Decompose $\log p(\mathbf{x}|\theta) = \mathcal{Q}(\theta;\theta^{(t)}) - \mathcal{H}(\theta;\theta^{(t)})$ where:

$$\mathcal{Q}(\theta;\theta^{(t)}) = \mathbb{E}_{p(\mathbf{z}|\mathbf{x},\theta^{(t)})}[\log p(\mathbf{x},\mathbf{z}|\theta)]$$

$$\mathcal{H}(\theta;\theta^{(t)}) = \mathbb{E}_{p(\mathbf{z}|\mathbf{x},\theta^{(t)})}[\log p(\mathbf{z}|\mathbf{x},\theta)]$$

The M-step ensures $\mathcal{Q}(\theta^{(t+1)};\theta^{(t)}) \geq \mathcal{Q}(\theta^{(t)};\theta^{(t)})$. By Gibbs' inequality (non-negativity of KL): $\mathcal{H}(\theta;\theta^{(t)}) \leq \mathcal{H}(\theta^{(t)};\theta^{(t)})$. Combining: $\log p(\mathbf{x}|\theta^{(t+1)}) \geq \log p(\mathbf{x}|\theta^{(t)})$. □

Missing data interpretation. EM handles three types of "missing data": (1) literally unobserved variables (sensor dropouts), (2) latent structure (cluster assignments), (3) hidden states in sequential models. The key requirement is that the complete-data log-likelihood $\log p(\mathbf{x},\mathbf{z}|\theta)$ must be easier to optimise than the observed-data log-likelihood $\log p(\mathbf{x}|\theta)$.

15.3 EM for Gaussian Mixture Models — Complete Derivation

Let K components with parameters $\square = \{\pi_k, \square_k, \Sigma_k\}_{k=1}^K$. Latent variable: $\mathbf{z}_i \in \{1,\dots,K\}$ (component assignment for example i).

Complete log-likelihood:

$$\log p(\mathbf{X},\mathbf{Z}|\square) = \sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}\left[\log \pi_k + \log \mathcal{N}(\mathbf{x}_i;\square_k,\Sigma_k)\right]$$

E-step. Compute posterior (responsibility):

$$r_{ik} = \mathbb{E}[z_{ik}|\mathbf{x}_i,\square^{(t)}] = \frac{\pi_k^{(t)}\mathcal{N}(\mathbf{x}_i;\square_k^{(t)},\Sigma_k^{(t)})}{\sum_j \pi_j^{(t)}\mathcal{N}(\mathbf{x}_i;\square_j^{(t)},\Sigma_j^{(t)})}$$

M-step. Maximise $Q(\square) = \mathbb{E}_{\mathbf{Z}|\mathbf{X},\square^{(t)}}[\log p(\mathbf{X},\mathbf{Z}|\square)]$:

$$\pi_k^{(t+1)} = \frac{1}{N}\sum_i r_{ik}, \qquad \square_k^{(t+1)} = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}, \qquad \Sigma_k^{(t+1)} = \frac{\sum_i r_{ik}(\mathbf{x}_i - \square_k^{(t+1)})(\mathbf{x}_i - \square_k^{(t+1)})^\top}{\sum_i r_{ik}}$$

M-step derivation for $\square_k$. Differentiate Q w.r.t. $\square_k$:

$$\frac{\partial Q}{\partial \square_k} = \sum_i r_{ik}\, \Sigma_k^{-1}(\mathbf{x}_i - \square_k) = \mathbf{0} \implies \square_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{\sum_i r_{ik}}$$

A responsibility-weighted mean: data points assigned more strongly to component k (higher r_{ik}) pull $\square_k$ toward them more.

EM as block-coordinate ascent. EM can be interpreted as alternating ascent on the ELBO, treating q and θ as separate variables:

- **E-step**: $q^{(t+1)} \leftarrow \arg\max_q \mathcal{L}_{\text{ELBO}}(q, \theta^{(t)}) = p_{\theta^{(t)}}(\mathbf{z}|\mathbf{x})$.
- **M-step**: $\theta^{(t+1)} \leftarrow \arg\max_\theta \mathcal{L}_{\text{ELBO}}(q^{(t+1)}, \theta) = \arg\max_\theta \mathbb{E}_{q^{(t+1)}}[\log p_\theta(\mathbf{x}, \mathbf{z})]$.

Since each step does not decrease the ELBO and the ELBO is bounded above, the sequence converges. Convergence is, however, only to stationary points (local optima or saddle points).

EM for GMMs: complete algorithm.

EM for Gaussian Mixture Models

E-step: compute the responsibilities

$$r_{ik} = \frac{\pi_k\, \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j\, \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

M-step: update the parameters

$$N_k = \sum_{i=1}^{N} r_{ik}, \quad \hat{\pi}_k = N_k/N$$

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{i=1}^{N} r_{ik}\mathbf{x}_i$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{1}{N_k} \sum_{i=1}^{N} r_{ik}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^\top$$

Convergence and practical issues. EM for GMMs can degenerate: if $\boldsymbol{\Sigma}_k \to 0$ for a component centred on a single data point, the likelihood diverges to ∞. Remedies include a conjugate prior on the precision matrix (variational EM) or a variance floor.

Generalised EM (GEM). It is not necessary to drive the M-step to full convergence: any update to θ that increases the ELBO is sufficient. Gradient ascent in the M-step (ECM) makes EM scalable to neural network parametrisations.

Online EM. For large N, the global E-step is expensive. *Online EM* updates sufficient statistics using a mini-batch and an exponential moving average with step size $\gamma_t \sim t^{-\alpha}$, $\alpha \in (0.5, 1)$.

Convergence guarantees. Under mild regularity conditions (the log-likelihood is bounded above and the EM mapping is continuous), the sequence $\{\theta^{(t)}\}$ produced by EM satisfies $\mathcal{L}(\theta^{(t+1)}) \geq \mathcal{L}(\theta^{(t)})$ (monotone increase) and every limit point is a stationary point of $\mathcal{L}$. For GMMs, the stationary points include saddle points and local maxima; EM does *not* guarantee the global maximum. In practice, multiple random restarts (different $\theta^{(0)}$) are used and the run with the highest final $\mathcal{L}$ is retained.

Number of components selection. The number of components K is a hyperparameter. Selection methods: (1) BIC penalty $\text{BIC}(K) = \mathcal{L}(\hat{\theta}_K) - \frac{d_K}{2} \log N$ where $d_K = K(d + d(d + 1)/2 + 1) - 1$ is the parameter count; (2) variational Bayes with a Dirichlet process prior (infinite GMM), which automatically sets irrelevant components to zero; (3) cross-validated held-out log-likelihood.

PART 16

Variational Inference

When the posterior $p(\mathbf{z} \mid \mathbf{x})$ is intractable, we approximate it with a simpler distribution $q_\phi(\mathbf{z})$ by maximising the ELBO. We derive CAVI and mean-field VI in full.

16.1 The Evidence Lower BOund (ELBO)

The ELBO is the heart of variational inference: it transforms the intracttabile di calcolare la distribuzione posteriore in un problema di ottimizzazione.

Problema di partenza. Dati dati $\mathbf{x}$ e variabili latenti $\mathbf{z}$, vogliamo la posteriore $p(\mathbf{z}|\mathbf{x})$. Per la regola di Bayes:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$$

The denominator $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\, d\mathbf{z}$ is in general intractable (integral over all of $\mathbb{R}^d$).

Derivation of the ELBO. Let $q_\phi(\mathbf{z}|\mathbf{x})$ be a parametric variational distribution (e.g. a diagonal Gaussian). The log-likelihood can be decomposed as:

$$\begin{aligned}
\log p_\theta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}, \mathbf{z})\, d\mathbf{z} \\
&= \log \int q_\phi(\mathbf{z}|\mathbf{x}) \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\, d\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\, d\mathbf{z} \quad \text{(Jensen, concavity of log)} \\
&= \mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}, \mathbf{z})] - \mathbb{E}_{q_\phi}[\log q_\phi(\mathbf{z}|\mathbf{x})]
\end{aligned}$$

$$\boxed{\text{ELBO}(\phi, \theta; \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))}$$

ELBO Decomposition

$$\underbrace{\log p_\theta(\mathbf{x})}_{\text{log-evidence}} = \underbrace{\text{ELBO}(\phi, \theta)}_{\text{optimised}} + \underbrace{D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x}))}_{\geq 0}$$

Since the KL is non-negative, ELBO $\leq \log p(\mathbf{x})$: the ELBO is a *lower bound* on the log-evidence. Maximising the ELBO is equivalent to:

1. Maximising $\mathbb{E}_q[\log p(\mathbf{x}|\mathbf{z})]$: increasing the reconstruction likelihood.
2. Minimising $D_{\text{KL}}(q \| p)$: keeping the variational distribution close to the prior.

EXAMPLE — ELBO for a GMM

Model: $z \sim \text{Cat}(\pi)$ (component), $x|z = k \sim \mathcal{N}(\mu_k, \Sigma_k)$. Prior: $p(z) = \text{Cat}(\pi_0)$.
Variational distribution: $q(z|x) = \text{Cat}(r)$ where $r_k \propto \pi_{0,k}\, \mathcal{N}(x; \mu_k, \Sigma_k)$.

$$\text{ELBO} = \sum_k r_k \log \mathcal{N}(x; \mu_k, \Sigma_k) - \sum_k r_k \log \frac{r_k}{\pi_{0,k}}$$

Maximising w.r.t. $\{r_k\}$ is the E-step of EM; w.r.t. $\{\mu_k, \Sigma_k, \pi_0\}$ is the M-step.

Amortised inference. In neural networks, the parameters ϕ of the encoder $q_\phi(\mathbf{z}|\mathbf{x})$ are shared across all samples: instead of optimising q separately for each $\mathbf{x}$, the network *amortises* inference by mapping $\mathbf{x} \mapsto (\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}_\phi(\mathbf{x}))$. This is the principle of the VAE (Section 17).

ELBO as a compression objective. The ELBO has an information-theoretic interpretation:

$$-\mathcal{L}_{\text{ELBO}} = \underbrace{-\mathbb{E}_q[\log p_\theta(\mathbf{x}|\mathbf{z})]}_{\text{reconstruction cost}} + \underbrace{D_{\text{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z})\big)}_{\text{latent coding cost}}$$

The first term measures how many bits are needed to reconstruct $\mathbf{x}$ given $\mathbf{z}$; the second measures how many bits to transmit $\mathbf{z}$ using the prior as a code. Minimising $-\mathcal{L}_{\text{ELBO}}$ finds the most compressed latent representation that still allows faithful reconstruction.

ELBO estimation: IWAE. Burda et al. (2016) propose the Importance Weighted Autoencoder (IWAE) with K samples:

$$\log p(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}_1,\ldots,\mathbf{z}_K \sim q_\phi}\left[\log \frac{1}{K}\sum_{k=1}^{K} \frac{p_\theta(\mathbf{x}, \mathbf{z}_k)}{q_\phi(\mathbf{z}_k|\mathbf{x})}\right] \triangleq \mathcal{L}_{\text{ELBO}\,K}$$

The sequence $\mathcal{L}_{\text{ELBO}1} \leq \mathcal{L}_{\text{ELBO}2} \leq \cdots \leq \log p(\mathbf{x})$ is monotone; for $K = 1$ it recovers the standard VAE.

β-VAE and disentanglement. Higgins et al. (2017) weight the KL term:

$$\mathcal{L}_\beta = \mathbb{E}_q[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta\, D_{\text{KL}}\big(q_\phi \,\|\, p\big)$$

For $\beta > 1$, stronger compression of the latent representation is enforced; empirically this promotes disentanglement (each dimension of $\mathbf{z}$ controls one independent factor of variation).

Three equivalent forms of the ELBO.

Form 1 (ELBO as lower bound):

$$\mathcal{L}_{\text{ELBO}} = \log p(\mathbf{x}) - D_{\text{KL}}(q(\mathbf{z}) \,\|\, p(\mathbf{z}|\mathbf{x})) \leq \log p(\mathbf{x})$$

Form 2 (expected log-joint minus entropy):

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z})] + H(q)$$

Form 3 (reconstruction minus KL):

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_q[\log p(\mathbf{x}|\mathbf{z})] - D_{\text{KL}}(q(\mathbf{z}) \,\|\, p(\mathbf{z}))$$

Form 3 is used in VAEs: the first term is reconstruction quality (negative decoder loss), the second is a regularisation term pulling the approximate posterior q toward the prior $p(\mathbf{z})$.

Tightness conditions. $\mathcal{L}_{\text{ELBO}} = \log p(\mathbf{x})$ iff $D_{\text{KL}}(q \,\|\, p(\mathbf{z}|\mathbf{x})) = 0$ iff $q = p(\mathbf{z}|\mathbf{x})$ exactly. For mean-field q, this is generally impossible unless the posterior factorises. The tightness gap quantifies how much information is lost by the mean-field approximation.

Multi-sample bound. The Importance-Weighted ELBO (IWAE):

$$\mathcal{L}_K = \mathbb{E}_{\mathbf{z}^{(1:K)} \sim q}\left[\log \frac{1}{K}\sum_{k=1}^{K} \frac{p(\mathbf{x}, \mathbf{z}^{(k)})}{q(\mathbf{z}^{(k)}|\mathbf{x})}\right]$$

satisfies $\mathcal{L}_1 = \mathcal{L}_{\text{ELBO}} \leq \mathcal{L}_2 \leq \cdots \leq \mathcal{L}_\infty = \log p(\mathbf{x})$. With $K = 50$, IWAE is significantly tighter than ELBO on standard benchmarks.

16.2 Mean-Field Variational Inference

The mean-field family. Restrict q to the class of fully factorised distributions:

$$q(\mathbf{z}) = \prod_{j=1}^{m} q_j(z_j)$$

Each factor q_j describes the marginal distribution of z_j independently of all other components. This is the most aggressive approximation possible: it eliminates all posterior dependencies.

Derivation of the optimal factor. Fixing all factors except q_j, the ELBO as a function of q_j becomes:

$$\begin{aligned}\mathcal{L}_{\text{ELBO}} &= \int q_j(z_j) \left\{ \mathbb{E}_{q_{-j}}[\log p(\mathbf{x}, \mathbf{z})] \right\} dz_j - \int q_j \log q_j \, dz_j + \text{const}_{-j} \\ &= -D_{\text{KL}}\big(q_j(z_j) \,\|\, \tilde{p}_j(z_j)\big) + \text{const}\end{aligned}$$

where the *pseudo-distribution* $\tilde{p}_j$ is defined by:

$$\log \tilde{p}_j(z_j) = \mathbb{E}_{q_{-j}}[\log p(\mathbf{x}, \mathbf{z})] + \text{const}$$

Mean-Field Update

The optimal factor (with all others fixed) is:

$$q_j^*(z_j) = \frac{\exp\big(\mathbb{E}_{q_{-j}}[\log p(\mathbf{x}, \mathbf{z})]\big)}{\int \exp\big(\mathbb{E}_{q_{-j}}[\log p(\mathbf{x}, \mathbf{z})]\big) \, dz_j}$$

Monotone convergence property. Each coordinate update for factor j does not decrease the ELBO: the KL w.r.t. $\tilde{p}_j$ is minimised to zero by setting $q_j = \tilde{p}_j$. Since the ELBO is bounded above by $\log p(\mathbf{x})$, the sequence of ELBO values converges. Convergence to the global optimum is not guaranteed: mean-field is a non-convex optimisation problem over $(q_1, \dots, q_m)$.

When is the mean-field family exact? Iff the latent variables are a posteriori independent, i.e. $p(\mathbf{z}|\mathbf{x}) = \prod_j p(z_j|\mathbf{x})$. For models with strong posterior dependencies (e.g. sequential models with temporally correlated latent variables), the mean-field approximation introduces a systematic error known as *mean-field bias*.

Example — Mean-field for a GMM (Gaussian Mixture Model)

Latent variables: assignment $c_i \in \{1, \dots, K\}$, means $\boldsymbol{\mu}_k$, precisions Λ_k. Factorisation: $q = \prod_i q(c_i) \cdot \prod_k q(\boldsymbol{\mu}_k, \Lambda_k)$.

- $q(c_i) = \text{Cat}(\mathbf{r}_i)$ with $r_{ik} \propto \exp(\mathbb{E}_q[\log \pi_k] + \frac{1}{2}\mathbb{E}_q[\log |\Lambda_k|] - \frac{d}{2}\log(2\pi) - \frac{1}{2}\mathbb{E}_q[(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top \Lambda_k (\mathbf{x}_i - \boldsymbol{\mu}_k)])$.
- $q(\boldsymbol{\mu}_k) = \mathcal{N}(\mathbf{m}_k, \mathbf{S}_k)$ with parameters updated according to the responsibilities r_{ik}.

This recovers the variational EM algorithm of Bishop (2006).

Limitations of mean-field. The optimal factor q_j^* must belong to a closed family for exact computation. For non-conjugate models (e.g. with non-conjugate priors or neural likelihoods), $\tilde{p}_j$ has no closed form. Solutions: (i) reparametrise toward the exponential family,

(ii) use Black-Box VI with stochastic gradients, or (iii) switch to the VAE architecture that directly parametrises q_ϕ with a neural network.

Restrict q_ϕ to fully factored (mean-field) distributions:

$$q(\mathbf{z}) = \prod_{j=1}^{m} q_j(z_j)$$

Each factor q_j is optimised independently. The optimal factor (holding all others fixed) is:

$$\log q_j^*(z_j) = \mathbb{E}_{q_{-j}}[\log p(\mathbf{x}, \mathbf{z})] + \text{const}$$

where q_{-j} denotes the product of all factors except j.

Derivation. The ELBO as a function of q_j alone:

$$\begin{aligned}\mathcal{L}_{\text{ELBO}} &= \int q_j(z_j) \left\{ \int_{-j} \prod_{k \neq j} q_k \, \log p(\mathbf{x}, \mathbf{z}) \, d\mathbf{z}_{-j} \right\} dz_j - \int q_j \log q_j \, dz_j + \text{const} \\ &= \mathbb{E}_{q_j}\left[\mathbb{E}_{q_{-j}}[\log p(\mathbf{x}, \mathbf{z})]\right] - \mathbb{E}_{q_j}[\log q_j] + \text{const} \\ &= -D_{\text{KL}}(q_j(z_j) \,\|\, \tilde{p}(z_j)) + \text{const}\end{aligned}$$

where $\tilde{p}(z_j) \propto \exp(\mathbb{E}_{q_{-j}}[\log p(\mathbf{x}, \mathbf{z})])$. Maximising over q_j gives $q_j^* = \tilde{p}$.

Mean-field ELBO. With $q(\mathbf{z}) = \prod_i q_i(z_i)$, the ELBO becomes:

$$\mathcal{L}_{\text{ELBO}} = \sum_i \mathbb{E}_{q_i}\left[\mathbb{E}_{\prod_{j \neq i} q_j}[\log p(\mathbf{x}, \mathbf{z})]\right] + \sum_i H(q_i)$$

Optimising over q_i while holding the other factors fixed gives the CAVI update (§16.3).

When does mean-field fail?

- *Bimodal posteriors:* mean-field forces q_i to be unimodal. For a bimodal posterior $p(z_1, z_2)$, the mean-field approximation places mass between the modes — an artefact.
- *Strong posterior correlations:* if z_1 and z_2 are highly correlated given $\mathbf{x}$, forcing $q(z_1, z_2) = q(z_1)q(z_2)$ severely misestimates the posterior covariance.
- *Hierarchical models with many layers:* errors compound across layers.

Structured mean-field. Allow some correlations: $q(\mathbf{z}) = \prod_g q_g(\mathbf{z}_g)$ where each group g may contain multiple variables. For example, in a time series, $q(z_{1:T}) = q(z_{1:T/2})q(z_{T/2+1:T})$ preserves within-block correlations. The tradeoff is tractability: each q_g must have a tractable form for the CAVI update.

16.3 CAVI: Coordinate Ascent Variational Inference

CAVI Algorithm

Input: data $\mathbf{x}$, model $p(\mathbf{x}, \mathbf{z})$, family $\{q_j\}$.
Initialise: q_j chosen randomly or from data moments.
Repeat until convergence:

$$\text{For } j = 1, \ldots, m: \quad \log q_j(z_j) \leftarrow \mathbb{E}_{q_{-j}}[\log p(\mathbf{x}, \mathbf{z})] + \text{const}$$

Compute $\mathcal{L}_{\text{ELBO}} = \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z})] - \mathbb{E}_q[\log q(\mathbf{z})]$ and check convergence.
Output: $q^* = \prod_j q_j^*$ as the posterior approximation.

Convergence guarantee. Let $\mathcal{L}_{\text{ELBO}}^{(t)}$ be the ELBO value at iteration t. The sequence $\{\mathcal{L}_{\text{ELBO}}^{(t)}\}$ is monotone non-decreasing and bounded above: it converges. The proof uses the fact that each CAVI update maximises the ELBO w.r.t. q_j with all others fixed (coordinate ascent = ascent by coordinates).

Per-iteration complexity.

- For conjugate exponential family models: each update requires $O(N \cdot d)$ operations (where N = data, d = latent dimension).
- For non-conjugate models: non-analytic expectations must be computed via Monte Carlo or linearisation.
- Computing the ELBO for convergence checking: $O(N \cdot m)$.

Stochastic CAVI (SVI — Stochastic Variational Inference). For large datasets, Hoffman et al. (2013) propose updating the local variables $\{q_j^{(i)}\}$ for a mini-batch $\mathcal{B} \subset \{1, \dots, N\}$ and using a stochastic gradient of the ELBO w.r.t. global parameters with a learning-rate schedule ρ_t satisfying the Robbins-Monro conditions ($\sum_t \rho_t = \infty$, $\sum_t \rho_t^2 < \infty$). Convergence to the same critical points as CAVI is guaranteed.

Example — CAVI for Bayesian regression

Model: $y_i = \mathbf{w}^\top \mathbf{x}_i + \epsilon_i$, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \tau^2 \mathbf{I})$.
Mean-field family: $q(\mathbf{w}) = \mathcal{N}(\mathbf{m}, \mathbf{S})$.
CAVI update:

$$\mathbf{S}^{-1} = \frac{1}{\sigma^2}\mathbf{X}^\top\mathbf{X} + \frac{1}{\tau^2}\mathbf{I}, \qquad \mathbf{m} = \frac{1}{\sigma^2}\mathbf{S}\mathbf{X}^\top\mathbf{y}$$

This has a closed-form solution (one iteration): it coincides with the exact posterior because the model is conjugate Gaussian.

CAVI vs. VAE. CAVI is appropriate when: (a) the model has closed-form optimal distributions q_j^*; (b) the dataset size is moderate. The VAE (Part 17) replaces the analytic update with a neural encoder $q_\phi(\mathbf{z}|\mathbf{x})$ and optimises ϕ by backpropagation, enabling generative models with non-conjugate likelihoods (e.g. images, text) on datasets of millions of examples.

Algorithm: CAVI

Initialise $\{q_j\}$ randomly.
Repeat until ELBO converges:
  For $j = 1, \dots, m$:

$$q_j(z_j) \leftarrow \frac{\exp\left(\mathbb{E}_{q_{-j}}[\log p(\mathbf{x}, \mathbf{z})]\right)}{\int \exp\left(\mathbb{E}_{q_{-j}}[\log p(\mathbf{x}, \mathbf{z})]\right) \, dz_j}$$

Compute $\mathcal{L}_{\text{ELBO}}$ and check convergence.

Convergence. Since each coordinate step increases $\mathcal{L}_{\text{ELBO}}$ (or leaves it unchanged) and $\mathcal{L}_{\text{ELBO}}$ is bounded above by $\log p(\mathbf{x})$, CAVI converges to a local optimum.

Convergence of CAVI. CAVI is guaranteed to converge to a local maximum of the ELBO

(since each update increases $\mathcal{L}_{\text{ELBO}}$ and the ELBO is bounded above by $\log p(\mathbf{x})$). However, CAVI can get trapped in poor local optima — especially for multimodal posteriors. Multiple random restarts help.

CAVI for a Gaussian model: worked example. Let $\mathbf{x}_n = \square + \epsilon_n$ with $\epsilon_n \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ and prior $\square \sim \mathcal{N}(\mathbf{0}, \sigma_0^2\mathbf{I})$. Mean-field: $q(\square, \{\square_n\}) = q(\square)\prod_n q(\square_n)$.

CAVI update for $q(\square)$:

$$\log q^*(\square) = \mathbb{E}_{-\square}[\log p(\mathbf{x}, \square, \ldots)] + \text{const}$$

For the Gaussian likelihood + Gaussian prior (both conjugate), this gives:

$$q^*(\square) = \mathcal{N}\left(\square;\ \frac{\sigma^{-2}\sum_n \mathbf{x}_n}{\sigma_0^{-2} + N\sigma^{-2}},\ \frac{1}{\sigma_0^{-2} + N\sigma^{-2}}\mathbf{I}\right)$$

This is the exact posterior (since the model is fully conjugate). CAVI is exact for models in the exponential family with conjugate priors.

Stochastic VI. For large datasets, CAVI's full-data expectation in the E-step is expensive. Stochastic VI (Hoffman et al., 2013) uses mini-batch estimates of the ELBO gradient, enabling online learning from data streams. Combined with natural gradient steps (using the Fisher metric), convergence is fast.

16.4 Black-Box Variational Inference

When conjugacy is unavailable (e.g. neural network models), the CAVI coordinate update has no closed form. Black-box VI (Ranganath et al. 2014) estimates the ELBO gradient using the *score function estimator* (also called REINFORCE gradient):

$$\nabla_\phi \mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\phi}\left[\nabla_\phi \log q_\phi(\mathbf{z}) \cdot \big(\log p(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z})\big)\right]$$

Derivation. Using the log-derivative trick: $\nabla_\phi q_\phi(\mathbf{z}) = q_\phi(\mathbf{z})\nabla_\phi \log q_\phi(\mathbf{z})$.

$$\nabla_\phi \mathcal{L}_{\text{ELBO}} = \nabla_\phi \int q_\phi(\mathbf{z}) \log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z})}\, d\mathbf{z} = \int \nabla_\phi q_\phi(\mathbf{z}) \cdot \log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z})}\, d\mathbf{z} - \int q_\phi(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z})\, d\mathbf{z}$$

The second integral $= \nabla_\phi \int q_\phi = 0$. Substituting the log-derivative trick into the first:

$$\nabla_\phi \mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\phi}\left[\nabla_\phi \log q_\phi(\mathbf{z}) \cdot \big(\log p(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z})\big)\right]$$

This is unbiased and requires only samples from q_ϕ and evaluations of $\log p(\mathbf{x}, \mathbf{z})$ — no differentiability required of the model. The cost is high variance: the factor $\log p(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z})$ can be very large or small, making single-sample estimates noisy. **Score function estimator (REINFORCE).** The ELBO gradient with respect to variational parameters ϕ is:

$$\nabla_\phi \mathcal{L}_{\text{ELBO}} = \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = \mathbb{E}_{q_\phi(\mathbf{z})}\left[f(\mathbf{z})\, \nabla_\phi \log q_\phi(\mathbf{z})\right]$$

where $f(\mathbf{z}) = \log p(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z})$. This identity follows from the log-derivative trick: $\nabla_\phi q_\phi = q_\phi \nabla_\phi \log q_\phi$. The resulting Monte Carlo estimator is unbiased:

$$\hat{g}_{\text{SF}} = \frac{1}{S}\sum_{s=1}^{S} f(\mathbf{z}^{(s)})\, \nabla_\phi \log q_\phi(\mathbf{z}^{(s)}), \quad \mathbf{z}^{(s)} \sim q_\phi$$

This is the *score function* (SF) or REINFORCE estimator. It works for any q_ϕ (discrete or continuous) without requiring differentiability of f.

Variance reduction via control variates. The SF estimator has high variance, making training slow. A *control variate* $c(\mathbf{z})$ reduces variance without introducing bias:

$$\hat{g}_{\text{CV}} = \frac{1}{S}\sum_{s}\left[f(\mathbf{z}^{(s)}) - c\right]\nabla_\phi \log q_\phi(\mathbf{z}^{(s)})$$

since $\mathbb{E}_q[c\,\nabla_\phi \log q_\phi] = c\nabla_\phi \int q_\phi\, d\mathbf{z} = c\nabla_\phi 1 = 0$. Common choices:

- *Constant baseline:* $c = \mathbb{E}_q[f(\mathbf{z})]$ estimated from the batch. Reduces variance proportionally to $\text{Var}(f)/(f-c)^2$ ratio.
- *Input-dependent baseline:* $c(\mathbf{x}) = b_\psi(\mathbf{x})$ learned by a separate network to minimise the variance of $\hat{g}$.
- *NVIL (Mnih & Gregor, 2014):* uses a learned baseline $c_\psi(\mathbf{x})$ plus variance normalisation to adaptively scale the gradient signal.

BBVI algorithm. The full Black-Box VI algorithm (Ranganath, Gerrish, Blei, 2014) is:

1. Initialise ϕ randomly.
2. For each iteration:
   (a) Sample $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(S)} \sim q_\phi(\mathbf{z}|\mathbf{x})$.
   (b) Compute $f^{(s)} = \log p(\mathbf{x}, \mathbf{z}^{(s)}) - \log q_\phi(\mathbf{z}^{(s)}|\mathbf{x})$ for each s.
   (c) Compute $\hat{g} = \frac{1}{S}\sum_s (f^{(s)} - \bar{f})\nabla_\phi \log q_\phi(\mathbf{z}^{(s)}|\mathbf{x})$ where $\bar{f} = \frac{1}{S}\sum f^{(s)}$.
   (d) Update: $\phi \leftarrow \phi + \eta\hat{g}$ (or Adam/RMSprop).
3. Return $q_{\hat{\phi}}$.

Example — BBVI vs reparameterisation: when to use which

Criterion	Score Function (BBVI)	Reparameterisation
Distribution family	Any (incl. discrete)	Must be reparametrisable
Variance	High (needs baselines)	Low
Implementation	Simple	Requires invertible noise
Applicability	Discrete latents, RL	Continuous latents (VAEs)

For continuous latents with Gaussian q_ϕ, reparameterisation is almost always preferred. For discrete latents (e.g. VQ-VAE, discrete VAE, text generation), SF estimators (with straight-through or REINFORCE) are the primary option.

ELBO pathwise gradient. When q_ϕ is reparametrisable — i.e. $\mathbf{z} = g_\phi(\square)$ with $\square \sim p(\square)$ fixed — the gradient becomes:

$$\nabla_\phi \mathcal{L}_{\text{ELBO}} = \mathbb{E}_{p(\square)}\left[\nabla_\phi f(g_\phi(\square))\right]$$

This pathwise (or reparameterisation) gradient has much lower variance because it differentiates through the sample path $g_\phi(\square)$ directly. For $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\square_\phi, \text{diag}(\square_\phi^2))$: $\mathbf{z} = \square_\phi + \square_\phi \odot \square$, $\square \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

16.5 Amortised Inference and the Inference Network

In standard VI, ϕ is optimised for each data point separately. **Amortised inference** (Kingma & Welling 2013) trains a shared inference network $q_\phi(\mathbf{z}|\mathbf{x})$ to approximate the posterior for all $\mathbf{x}$ simultaneously:

$$\phi^* = \arg\max_\phi \mathbb{E}_{p_{\text{data}}(\mathbf{x})}[\mathcal{L}_{\text{ELBO}}(\theta, \phi; \mathbf{x})]$$

The inference network "amortises" the cost of inference across the dataset: after training, posterior approximation for a new $\mathbf{x}$ requires only a forward pass through the encoder. This is the core idea of the VAE (Part 17).

Trade-off. Amortisation gap: since the inference network must generalise across all $\mathbf{x}$, $q_{\phi^*}(\mathbf{z}|\mathbf{x})$ may not be as accurate as an $\mathbf{x}$-specific optimised $q_\phi^{\mathbf{x}}$. The amortisation gap $\text{AG}(\mathbf{x}) = \mathcal{L}_{\text{ELBO}}(\phi_{\mathbf{x}}^*) - \mathcal{L}_{\text{ELBO}}(\phi^*)$ measures this loss. Semi-amortised inference (Kim et al., 2018) fine-tunes ϕ per test point starting from the encoder output, combining speed and accuracy.

Expressiveness of q. Mean-field $q_\phi(\mathbf{z}|\mathbf{x}) = \prod_i q_i(z_i|\mathbf{x})$ misses posterior correlations. Improving the approximate posterior:

- *Normalising flows:* apply a sequence of invertible transforms $\mathbf{z} = f_K \circ \cdots \circ f_1(\mathbf{z}_0)$, $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, to obtain a richer posterior family (Rezende & Mohamed, 2015).
- *Importance-weighted ELBO:* $\text{IWAE}_K = \mathbb{E}_{\mathbf{z}^{(1:K)}\sim q}\left[\log \frac{1}{K}\sum_{k=1}^K \frac{p(\mathbf{x},\mathbf{z}^{(k)})}{q(\mathbf{z}^{(k)}|\mathbf{x})}\right] \geq \mathcal{L}_{\text{ELBO}}$. Tighter bound with more samples; converges to $\log p(\mathbf{x})$ as $K \to \infty$ (Burda et al., 2016).

the *amortisation gap*: the optimised per-datapoint $\phi^*(\mathbf{x})$ is always at least as good as the shared ϕ^*. In practice, the gap is small and the computational savings enormous.

Semi-amortised inference. Kim et al. (2018) propose a hybrid: use the encoder as an initialisation, then run a few steps of stochastic gradient ascent on the ELBO w.r.t. local variational parameters λ_n initialised at $\phi(\mathbf{x}_n)$. This closes much of the amortisation gap at a modest computational cost ($\approx$ 5–10 extra gradient steps per example at test time).

Implicit variational distributions. Instead of parameterising $q_\phi(\mathbf{z}|\mathbf{x})$ as a known distribution family, one can define q_ϕ implicitly via a generator:

$$\mathbf{z} = G_\phi(\mathbf{x}, \square), \quad \square \sim p(\square)$$

where G_ϕ is any differentiable function. This defines a distribution over $\mathbf{z}$ without specifying its density in closed form. The reparameterisation trick still applies; the KL term, however, requires density-ratio estimation (e.g. via a discriminator as in adversarial autoencoders, Makhzani et al., 2015) rather than analytic computation.

Aggregated posterior and marginal matching. The aggregated posterior $q_\phi(\mathbf{z}) = \mathbb{E}_{p_{\text{data}}(\mathbf{x})}[q_\phi(\mathbf{z}|\mathbf{x})]$ should match the prior $p(\mathbf{z})$ for the VAE to be a good generative model. If $q_\phi(\mathbf{z}) \neq p(\mathbf{z})$, sampling $\mathbf{z} \sim p(\mathbf{z})$ and decoding gives samples outside the encoder's posterior support — the classic "holes in the latent space" problem. Regularising $D_{\text{KL}}\big(q_\phi(\mathbf{z}) \,\|\, p(\mathbf{z})\big)$ directly (e.g. via MMD in the InfoVAE, Zhao et al., 2019) addresses this at the cost of slightly more complex training.

PART 17

Variational Autoencoder (VAE)

We derive the VAE from scratch: the generative model, the inference network, the reparameterisation trick, and the full ELBO loss. Every gradient is derived explicitly.

Amortisation gap. Amortised inference approximates the optimal per-datapoint posterior $q^*(\mathbf{z}|\mathbf{x}_n)$ with a shared function $q_\phi(\mathbf{z}|\mathbf{x}_n)$. The *amortisation gap* is:

$$\mathcal{L}^*_{\text{ELBO}} - \mathcal{L}_{\text{ELBO}\,\phi} = \mathbb{E}_n\left[D_{\text{KL}}(q^*(\mathbf{z}|\mathbf{x}_n) \,\|\, p(\mathbf{z}|\mathbf{x}_n)) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}_n) \,\|\, p(\mathbf{z}|\mathbf{x}_n))\right] \geq 0$$

This gap measures how much the inference network hurts the ELBO relative to the optimal per-point variational distribution. It vanishes when the inference network is expressive enough and well-trained. Semi-amortised inference (Kim et al., 2018) mitigates the gap by taking a few gradient steps of local optimisation from the amortised initialisation.

Encoder architecture choices. The inference network $q_\phi(\mathbf{z}|\mathbf{x})$ must map diverse data $\mathbf{x}$ to distribution parameters $(\square, \log \square)$ of q_ϕ. Choices:

- *MLP encoder:* for tabular/vector data. Output: $[\square_\phi(\mathbf{x}), \log \square_\phi(\mathbf{x})] = f_\phi(\mathbf{x})$.
- *CNN encoder:* for images. Typical architecture: Conv→BN→ReLU repeated 3–5 times, then global average pooling, then linear layer to $(\square, \log \square)$.
- *ResNet encoder:* hierarchical representation for high-resolution images. Used in VQVAE-2, DALL-E.
- *Transformer encoder:* for sequences (text, time series). $\mathbf{x}$ tokens → [CLS] representation → linear to $(\square, \log \square)$.

Normalising flow posteriors. The mean-field Gaussian q_ϕ may be too restricted for complex posteriors. Normalising flows augment q_ϕ with K invertible transformations:

$$\mathbf{z}_0 \sim \mathcal{N}(\square_\phi, \text{diag}(\square^2_\phi)), \quad \mathbf{z}_K = T_K \circ \cdots \circ T_1(\mathbf{z}_0)$$

$$\log q_\phi(\mathbf{z}_K|\mathbf{x}) = \log q_0(\mathbf{z}_0|\mathbf{x}) - \sum_{k=1}^{K} \log |\det J_{T_k}(\mathbf{z}_{k-1})|$$

Used in IAF-VAE (Kingma et al., 2016) which uses Inverse Autoregressive Flows for expressive posteriors with $O(d)$ cost (parallel at inference time, since the flow conditions on $\mathbf{z}_0$ not on previous $\mathbf{z}_k$).

17.1 The Generative Model

The Variational Autoencoder (VAE, Kingma & Welling, 2014) is the first deep generative model trained end-to-end with backpropagation. It combines a latent generative model with an encoder-decoder architecture.

VAE — Generative Model

The generative model (decoder) is:

$$\mathbf{z} \sim p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I}), \qquad \mathbf{x}|\mathbf{z} \sim p_\theta(\mathbf{x}|\mathbf{z})$$

The conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$ is parametrised by a neural network:

- For binary images: $p_\theta(\mathbf{x}|\mathbf{z}) = \text{Ber}(\boldsymbol{\mu}_\theta(\mathbf{z}))$ (Bernoulli)

- For continuous images: $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{z}), \sigma^2\mathbf{I})$

Latent space intuition. The prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$ forces the latent space to be *regular*: every point in the multi-dimensional unit sphere decodes into a plausible sample. The latent dimension d_z controls compression: $d_z = 2$ for visualisation, $d_z \in [64, 256]$ typically for images.

Why not a standard autoencoder? A deterministic autoencoder can "collapse": it maps each input to a precise point in the latent space without regularity. Sampling a random latent point then produces incoherent noise. The VAE forces the posterior $q_\phi(\mathbf{z}|\mathbf{x})$ to remain close to the Gaussian prior, guaranteeing regularity and interpolability.

Example — Generation with a trained VAE

To generate a new image:

1. Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ with $d_z = 64$.
2. Pass to the decoder: $\boldsymbol{\mu} = f_\theta(\mathbf{z}) \in [0,1]^{28\times 28}$ (for MNIST).
3. The generated image is $\boldsymbol{\mu}$ (or a Bernoulli sample from it).

For interpolation between two images $\mathbf{x}_1$ and $\mathbf{x}_2$:

$$\mathbf{z}_\lambda = (1-\lambda)\boldsymbol{\mu}_\phi(\mathbf{x}_1) + \lambda\boldsymbol{\mu}_\phi(\mathbf{x}_2), \quad \lambda \in [0,1]$$

The intermediate images $f_\theta(\mathbf{z}_\lambda)$ show a semantically coherent transition.

Decoder architectures. The choice of decoder $p_\theta(\mathbf{x}|\mathbf{z})$ determines the reconstruction loss:

- *Gaussian decoder:* $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mu_\theta(\mathbf{z}), \sigma^2\mathbf{I})$. Reconstruction loss $= \|\mathbf{x} - \mu_\theta(\mathbf{z})\|^2/(2\sigma^2)$ — MSE. Tends to produce blurry images (MSE averages over modes).
- *Bernoulli decoder* (binary images): $p_\theta(\mathbf{x}|\mathbf{z}) = \prod_i \text{Bernoulli}(x_i; \sigma(\mu_\theta(\mathbf{z})_i))$. Reconstruction loss = BCE. Used for binarised MNIST.
- *Discretised Gaussian / logistic mixture* (natural images): models pixel intensities in $\{0, \dots, 255\}$ with a mixture of logistics. Used in PixelCNN++ and VQ-VAE.

The blurriness problem and VQ-VAE. Standard VAEs produce blurry samples because: (1) MSE reconstruction averages over the posterior, and (2) the Gaussian prior is smooth. VQ-VAE (van den Oord et al., 2017) replaces the continuous latent with a *discrete* codebook $\{\mathbf{e}_k\}_{k=1}^K$:

$$\mathbf{z}_q = \mathbf{e}_{k^*}, \quad k^* = \arg\min_k \|\mathbf{z}_e - \mathbf{e}_k\|$$

Gradients flow through the quantisation via the straight-through estimator: $\nabla \approx \nabla_{\mathbf{z}_q}$. VQ-VAE encodes images into a discrete token grid; a powerful autoregressive prior (PixelCNN, Transformer) generates sequences of tokens. This is the basis of DALL-E v1 and many image tokenisation approaches.

Decoder architecture choices. The conditional $p_\theta(\mathbf{x}|\mathbf{z})$ is a neural network mapping $\mathbf{z} \in \mathbb{R}^d$ to distribution parameters over $\mathbf{x} \in \mathbb{R}^D$. Common choices:

- *Bernoulli decoder* (binary data): $p_\theta(\mathbf{x}|\mathbf{z}) = \prod_i \text{Bern}(x_i; \sigma(f_\theta(\mathbf{z})_i))$. Log-likelihood is binary cross-entropy.
- *Gaussian decoder* (continuous data): $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \square_\theta(\mathbf{z}), \sigma^2\mathbf{I})$. Log-likelihood is MSE (up to constants). Usually σ^2 is fixed (e.g. $\sigma^2 = 1$) to avoid over-fitting the variance.

- *Laplace decoder*: heavier tails than Gaussian, equivalent to L1 reconstruction loss. Better for sparse signals.
- *Mixture of Gaussians decoder*: $p_\theta(\mathbf{x}|\mathbf{z}) = \sum_k \pi_k(\mathbf{z})\mathcal{N}(\mathbf{x}; \square_k(\mathbf{z}), \Sigma_k(\mathbf{z}))$. More expressive but harder to train.
- *Discretised logistic* (PixelVAE, VQVAE): for 8-bit images, model each pixel channel as a discretised logistic distribution. Sharper reconstructions than Gaussian.

The prior $p(\mathbf{z})$. The standard choice is $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. The KL term in the ELBO penalises the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ for deviating from the prior. Alternative priors:

- *Mixture of Gaussians prior:* $p(\mathbf{z}) = \sum_k \pi_k \mathcal{N}(\mathbf{z}; \square_k, \Sigma_k)$. Encourages multimodal structure in the latent space.
- *VampPrior* (Tomczak & Welling, 2018): $p(\mathbf{z}) = \frac{1}{K}\sum_k q_\phi(\mathbf{z}|\mathbf{u}_k)$ where $\{\mathbf{u}_k\}$ are learned pseudo-inputs. Adapts the prior to the data manifold.
- *Flow-based prior:* $\mathbf{z} = T(\square), \square \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ where T is a normalising flow. Enables arbitrarily complex priors while maintaining tractable sampling.

17.2 The Inference Network and ELBO

The encoder as a variational approximator. In the VAE, the inference network (encoder) $q_\phi(\mathbf{z}|\mathbf{x})$ is a neural network that maps each input $\mathbf{x}$ to the parameters of a diagonal Gaussian:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}\big(\mathbf{z};\, \boldsymbol{\mu}_\phi(\mathbf{x}),\, \mathrm{diag}(\boldsymbol{\sigma}^2_\phi(\mathbf{x}))\big)$$

The encoder produces two vectors: $\boldsymbol{\mu}_\phi(\mathbf{x}) \in \mathbb{R}^{d_z}$ and $\log \boldsymbol{\sigma}^2_\phi(\mathbf{x}) \in \mathbb{R}^{d_z}$ (log is preferred for numerical stability and to guarantee $\sigma^2 > 0$).

ELBO for the VAE. Applying the decomposition from Section 15.1:

$$\mathcal{L}_{\text{ELBO}}(\phi, \theta; \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{\text{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z})\big)$$

Analytic KL. For the choice $q_\phi = \mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$ and prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$:

$$D_{\text{KL}}(q_\phi \,\|\, p) = \frac{1}{2}\sum_{j=1}^{d_z}\left(\mu_j^2 + \sigma_j^2 - 1 - \log \sigma_j^2\right)$$

This has analytic gradients w.r.t. $\boldsymbol{\mu}_\phi$ and $\boldsymbol{\sigma}^2_\phi$:

$$\frac{\partial D_{\text{KL}}(\,\|\,}{\partial \mu_j} = \mu_j, \qquad \frac{\partial D_{\text{KL}}(\,\|\,}{\partial \sigma_j^2} = \frac{1}{2}\left(1 - \frac{1}{\sigma_j^2}\right)$$

VAE Loss (single data point)

$$\mathcal{L}(\phi, \theta; \mathbf{x}) = -\mathcal{L}_{\text{ELBO}} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}_{\text{reconstruction loss}} + \underbrace{D_{\text{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z})\big)}_{\text{regularisation}}$$

Monte Carlo estimate of the reconstruction term. The expectation $\mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}|\mathbf{z})]$ is estimated with a single sample (after reparametrisation, Section 17.3):

$$\mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}|\mathbf{z})] \approx \log p_\theta(\mathbf{x}|\hat{\mathbf{z}}), \qquad \hat{\mathbf{z}} \sim q_\phi(\cdot|\mathbf{x})$$

The variance of this estimator is low because $\log p_\theta(\mathbf{x}|\mathbf{z})$ is a relatively smooth function of $\mathbf{z}$ (the decoder network is smooth). In practice $L = 1$ or $L = 5$ samples are used per mini-batch of 128 examples.

Introduce an *encoder* (recognition network) $q_\phi(\mathbf{z} \mid \mathbf{x})$ parameterised by ϕ:

$$q_\phi(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}\big(\mathbf{z};\, \square_\phi(\mathbf{x}),\, \mathrm{diag}(\square_\phi^2(\mathbf{x}))\big)$$

From the ELBO in form (**??**):

$$\mathcal{L}_{\mathrm{ELBO}}(\theta, \phi; \mathbf{x}) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z} \mid \mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]}_{\text{reconstruction}} - \underbrace{D_{\mathrm{KL}}\big(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z})\big)}_{\text{regularisation}}$$

KL term in closed form. With $q_\phi = \mathcal{N}(\square_\phi, \mathrm{diag}(\square_\phi^2))$ and $p = \mathcal{N}(\mathbf{0}, \mathbf{I})$:

$$D_{\mathrm{KL}}(q_\phi \,\|\, p) = -\frac{1}{2} \sum_{j=1}^{d} \left(1 + \log \sigma_{\phi,j}^2 - \mu_{\phi,j}^2 - \sigma_{\phi,j}^2\right)$$

This is differentiable with respect to ϕ and has no stochasticity.

Posterior collapse. A pathology of VAE training: some or all latent dimensions z_j become independent of $\mathbf{x}$ (the encoder ignores $\mathbf{x}$ for those dimensions, setting $\mu_j \approx 0$, $\sigma_j \approx 1$). The KL term drives these dimensions to the prior; the decoder learns to ignore them. Formally, dimension j has collapsed when $D_{\mathrm{KL}}\big(q_\phi(z_j|\mathbf{x}) \,\|\, p(z_j)\big) \approx 0$ for all $\mathbf{x}$.

Causes: (1) A sufficiently expressive decoder can model $p_\theta(\mathbf{x})$ without using latent information (autoregressive decoders are especially prone). (2) The KL regularisation dominates early in training before the reconstruction term.

Remedies:

- **β-VAE** (Higgins et al., 2017): weight the KL term by $\beta < 1$ initially and anneal to $\beta = 1$:
$$\mathcal{L}_{\beta\text{-VAE}} = -\mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}|\mathbf{z})] + \beta\, D_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z})\big)$$
$\beta > 1$ encourages disentangled representations; $\beta < 1$ (during warm-up) prevents collapse.
- **KL annealing**: linearly increase β from 0 to 1 over the first K training steps.
- **Free bits** (Kingma et al., 2016): set $\mathcal{L}_{\mathrm{KL}} = \max(\lambda, D_{\mathrm{KL}}(q_\phi \,\|\, p))$ per dimension; this guarantees at least λ nats of information per latent.

Importance-Weighted ELBO (IWAE). The ELBO is a lower bound on $\log p_\theta(\mathbf{x})$; it is tight only when $q_\phi(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$. Burda et al. (2016) derive a tighter bound using K samples:

$$\mathcal{L}_K = \mathbb{E}_{\mathbf{z}^{(1:K)} \sim q_\phi(\cdot|\mathbf{x})}\left[\log \frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(\mathbf{x}, \mathbf{z}^{(k)})}{q_\phi(\mathbf{z}^{(k)}|\mathbf{x})}\right] \leq \log p_\theta(\mathbf{x})$$

- $\mathcal{L}_1 = \mathcal{L}_{\mathrm{ELBO}}$ (standard ELBO).
- $\mathcal{L}_K$ is non-decreasing in K: more samples $\Rightarrow$ tighter bound.
- $\lim_{K \to \infty} \mathcal{L}_K = \log p_\theta(\mathbf{x})$.

The gradient of $\mathcal{L}_K$ with respect to ϕ uses self-normalised importance weights $\tilde{w}_k = w_k / \sum_j w_j$ where $w_k = p_\theta(\mathbf{x}, \mathbf{z}^{(k)}) / q_\phi(\mathbf{z}^{(k)}|\mathbf{x})$:

$$\nabla_\phi \mathcal{L}_K = \sum_{k=1}^{K} \tilde{w}_k \, \nabla_\phi \log q_\phi(\mathbf{z}^{(k)}|\mathbf{x})$$

This is equivalent to a weighted REINFORCE estimator. With reparameterisation, the path-gradient estimator $\nabla_\phi \mathcal{L}_K = \mathbb{E}[\sum_k \tilde{w}_k \nabla_\phi \log(w_k)]$ has lower variance.

Practical guidance for encoder design. Several choices strongly affect VAE performance:

Choice	Effect	Recommendation
Latent dim d_z	Larger: more capacity; less disentangled	64–256 for images
$\log \sigma$ output	Numerically stable	Always use $\log \sigma^2$, not σ
KL weight β	$\beta > 1$: more disentangled	Anneal $0 \to 1$ for first 10% of training
L samples	Higher L: lower variance	$L = 1$ with reparam. is usually sufficient

17.3 The Reparameterisation Trick

The problem. We want to compute $\nabla_\phi \, \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})]$ where f is typically $\log p_\theta(\mathbf{x}|\mathbf{z})$. Writing

$$\nabla_\phi \, \mathbb{E}_{q_\phi}[f] = \nabla_\phi \int f(\mathbf{z}) \, q_\phi(\mathbf{z}|\mathbf{x}) \, d\mathbf{z}$$

is not directly computable: sampling from q_ϕ breaks the computational graph and the gradient cannot flow through the sampling operation.

Solution: reparametrisation. Write $\mathbf{z}$ as a deterministic function of ϕ and a fixed-distribution noise $\boldsymbol{\epsilon}$:

$$\mathbf{z} = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\sigma}_\phi(\mathbf{x}) \odot \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

where $\odot$ is the Hadamard product and $\boldsymbol{\sigma}_\phi = \exp(\frac{1}{2} \log \boldsymbol{\sigma}^2_\phi)$.

Reparametrisation Trick Gradient

$$\nabla_\phi \, \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] = \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\nabla_\phi f\left(\boldsymbol{\mu}_\phi + \boldsymbol{\sigma}_\phi \odot \boldsymbol{\epsilon} \right) \right]$$

The gradient $\nabla_\phi f(\mathbf{z})$ is computed via backpropagation, and the expectation is estimated with a single sample of $\boldsymbol{\epsilon}$.

Comparison with REINFORCE. The classical alternative is the REINFORCE estimator (score function estimator, Williams 1992):

$$\nabla_\phi \, \mathbb{E}_{q_\phi}[f] = \mathbb{E}_{q_\phi} \left[f(\mathbf{z}) \, \nabla_\phi \log q_\phi(\mathbf{z}|\mathbf{x}) \right]$$

This is unbiased but has high variance ($\approx 10^2$–$10^4\times$ the variance of the reparametrisation trick). Reparametrisation reduces variance by exploiting the Gaussian structure and computing the gradient directly through $\mathbf{z}(\boldsymbol{\epsilon}; \phi)$.

Generalisations.

- **Cauchy distribution**: $z = \mu + \sigma \tan(\pi(u - \frac{1}{2}))$ with $u \sim \text{Uniform}(0, 1)$.
- **Gamma distribution**: Dasgupta & Langmead (2020) propose a reparametrisation via a differentiable rejection sampler.

- **Discrete variables**: the categorical distribution is not exactly reparametrisable; approximations such as *Gumbel-Softmax* (Jang et al., 2017) are used: $z_k \propto \exp((\log \pi_k + g_k)/\tau)$ with $g_k \sim \text{Gumbel}(0, 1)$ and temperature $\tau \to 0$ for the hard limit.
- **Normalising flows**: reparametrisation also applies to flow VAEs where $\mathbf{z} = f_\phi(\boldsymbol{\epsilon})$ with f_ϕ invertible (Part 18).

Example — Gradient computation for the reconstruction term

Let $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; f_\theta(\mathbf{z}), \sigma_x^2 \mathbf{I})$. Then:

$$\log p_\theta(\mathbf{x}|\mathbf{z}) = -\frac{\|\mathbf{x} - f_\theta(\mathbf{z})\|^2}{2\sigma_x^2} + \text{const}$$

With $\hat{\mathbf{z}} = \boldsymbol{\mu}_\phi + \boldsymbol{\sigma}_\phi \odot \hat{\boldsymbol{\epsilon}}$:

$$\nabla_\phi \log p_\theta(\mathbf{x}|\hat{\mathbf{z}}) = \frac{1}{\sigma_x^2}(f_\theta(\hat{\mathbf{z}}) - \mathbf{x})^\top \nabla_\phi f_\theta(\hat{\mathbf{z}})$$

The term $\nabla_\phi f_\theta(\hat{\mathbf{z}})$ is computed via backpropagation through the decoder f_θ and then through $\hat{\mathbf{z}} = \boldsymbol{\mu}_\phi + \boldsymbol{\sigma}_\phi \odot \hat{\boldsymbol{\epsilon}}$.

The reconstruction term requires computing $\mathbb{E}_{q_\phi(\mathbf{z}\,|\,\mathbf{x})}[\log p_\theta(\mathbf{x}\,|\,\mathbf{z})]$. To backpropagate through the expectation over $\mathbf{z}$, we need ∇_ϕ, but $\mathbf{z} \sim q_\phi$ depends on ϕ.

Naive approach (fails): $\nabla_\phi \mathbb{E}_{q_\phi}[f(\mathbf{z})] = \int f(\mathbf{z}) \nabla_\phi q_\phi(\mathbf{z})\, d\mathbf{z}$ — this requires sampling from $\nabla_\phi q_\phi$, which is not a distribution.

Reparameterisation. Sample $\square \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and set:

$$\mathbf{z} = \square_\phi(\mathbf{x}) + \square_\phi(\mathbf{x}) \odot \square$$

Now $\mathbf{z}$ is a deterministic function of ϕ and $\square$. The gradient flows through $\square_\phi$ and $\square_\phi$:

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}\,|\,\mathbf{x})}[f(\mathbf{z})] = \mathbb{E}_{\square \sim \mathcal{N}(\mathbf{0},\mathbf{I})}\left[\nabla_\phi f(\square_\phi + \square_\phi \odot \square)\right]$$

Estimated by L Monte Carlo samples: $\approx \frac{1}{L}\sum_{l=1}^{L} \nabla_\phi f(\mathbf{z}^{(l)})$ with $\mathbf{z}^{(l)} = \square_\phi + \square_\phi \odot \square^{(l)}$.

Geometric intuition. The reparameterisation separates randomness ($\square$, not differentiable w.r.t. ϕ) from the deterministic transform ($\square_\phi, \square_\phi$, differentiable). The gradient computation can "look through" the noise.

General reparameterisation. For a family $q_\phi(\mathbf{z}|\mathbf{x})$, if there exists a differentiable map g_ϕ and noise $\epsilon \sim p(\epsilon)$ (independent of ϕ) such that $\mathbf{z} = g_\phi(\epsilon, \mathbf{x})$, then:

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] = \mathbb{E}_{p(\epsilon)}\left[\nabla_\phi f(g_\phi(\epsilon, \mathbf{x}))\right]$$

The gradient passes *through* g_ϕ by the chain rule. This is valid for any differentiable g_ϕ.

Comparison with score function estimator.

Estimator	Bias	Variance
Differentiability		
Score function (REINFORCE)	Unbiased	High
Not required		
Reparameterisation	Unbiased	Low
Required		
Straight-through	Biased	Low
Not required		

Reparameterisation is preferred when g_ϕ is differentiable (continuous latents). For discrete latents, the straight-through estimator or Gumbel-Softmax relaxation is used.

Gumbel-Softmax. To reparameterise a categorical $z \sim \text{Cat}(\mathbf{p})$: sample $g_k = -\log(-\log u_k)$ where $u_k \sim \text{Uniform}(0,1)$ (Gumbel noise). Then:

$$\tilde{z}_k = \frac{\exp((g_k + \log p_k)/\tau)}{\sum_j \exp((g_j + \log p_j)/\tau)}$$

As $\tau \to 0$, $\tilde{\mathbf{z}} \to$ one-hot$(\arg\max_k(g_k + \log p_k))$ — the Gumbel-max trick. For finite τ, $\tilde{\mathbf{z}}$ is a continuous, differentiable approximation. Used in VQ-VAE training and discrete latent variable models.

17.4 Complete VAE Training Step

Algorithm: VAE Training Step

Given data point $\mathbf{x}$:

1. **Encode:** compute $\square_\phi(\mathbf{x})$, $\square^2_\phi(\mathbf{x})$ via encoder network.
2. **Sample:** $\square \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$; $\mathbf{z} = \square_\phi + \square_\phi \odot \square$.
3. **Decode:** compute $\square_\theta(\mathbf{z})$, $\square^2_\theta(\mathbf{z})$ via decoder.
4. **Loss:**
$$\mathcal{L} = -\mathbb{E}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] + D_{\text{KL}}\big(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z})\big)$$
5. **Backpropagate** through steps 3–1 using the reparameterisation for step 2.

Example — VAE loss at one data point

Suppose $d = 2$, encoder outputs $\square_\phi = (0.3, -0.1)$, $\square^2_\phi = (0.9, 1.1)$.
KL term:

$$D_{\text{KL}}\big(q_\phi \,\|\, p\big) = -\tfrac{1}{2}\left[(1 + \log 0.9 - 0.09 - 0.9) + (1 + \log 1.1 - 0.01 - 1.1)\right]$$

$$= -\tfrac{1}{2}\left[(1 - 0.1054 - 0.09 - 0.9) + (1 + 0.0953 - 0.01 - 1.1)\right]$$

$$= -\tfrac{1}{2}\left[-0.0954 + (-0.0147)\right] = -\tfrac{1}{2}(-0.1101) = 0.0551$$

Sample $\square = (0.5, -0.8)$: $\mathbf{z} = (0.3 + 0.949 \cdot 0.5,\ -0.1 + 1.049 \cdot (-0.8)) = (0.774,\ -0.939)$.
Reconstruction loss depends on decoder output and observed $\mathbf{x}$; the total VAE loss is their sum.

PART 18

Normalising Flows

Normalising flows transform a simple distribution into a complex one via a sequence of invertible, differentiable maps. We derive the change-of-variables formula and build autoregressive flows.

Choice of reconstruction likelihood.

- **Binary images (MNIST)**: $p_\theta(\mathbf{x}|\mathbf{z}) = \prod_i \text{Ber}(x_i; \sigma(f_{\theta,i}(\mathbf{z})))$. Loss = binary cross-entropy.
- **Continuous images**: $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; f_\theta(\mathbf{z}), \sigma_x^2 \mathbf{I})$. Loss = scaled MSE.
- **Discrete images (256 levels)**: $p_\theta(x_i|\mathbf{z}) = \text{Cat}(\text{softmax}(f_{\theta,i}(\mathbf{z})))$. Loss = discrete cross-entropy. Used in VQ-VAE.

Typical architectures.

- **MLP encoder** (tabular data): $\mathbb{R}^d \to \mathbb{R}^{512} \to \mathbb{R}^{256} \to \mathbb{R}^{2d_z}$ (mean and log-variance).
- **CNN encoder** (images): stack of Conv+BatchNorm+ReLU, then FC $\to 2d_z$.
- **CNN decoder**: FC $\to$ stack of ConvTranspose (deconvolutions) to reconstruct the image.
- **Transformer encoder** (text, sequences): multi-head attention + pooling $\to 2d_z$.

Critical hyperparameters.

- d_z: latent space dimension. Higher = less compression = more faithful reconstruction but less structured space.
- β: weight of the KL term. $\beta = 1$ (standard VAE), $\beta > 1$ (β-VAE for disentanglement).
- Learning rate: Adam with $\eta = 10^{-3}$, batch size 128–512.

Training monitoring. Track reconstruction loss and KL separately to diagnose:

- KL ≈ 0 for all dimensions $\Rightarrow$ posterior collapse: the encoder ignores $\mathbf{x}$, $q \approx p$.
- Very high KL $\Rightarrow$ bottleneck too narrow; increase d_z or reduce β.
- Stagnant reconstruction loss $\Rightarrow$ decoder too weak or d_z too small.

The β-VAE. Higgins et al. (2017) introduce a weight $\beta > 1$ on the KL term:

$$\mathcal{L}_{\beta\text{-VAE}} = -\mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}|\mathbf{z})] + \beta\, D_{\text{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z})\big)$$

$\beta > 1$ forces stronger disentanglement: each latent dimension z_i is encouraged to encode a single independent factor of variation (the KL constraint is tighter, pushing toward the isotropic Gaussian prior which has no correlations). The tradeoff: higher β degrades reconstruction quality. β-VAE achieves disentangled representations on dSprites, 3D shapes, and CelebA.

Posterior collapse. A pathology of VAEs: the decoder $p_\theta(\mathbf{x}|\mathbf{z})$ ignores $\mathbf{z}$ entirely, and the encoder collapses to the prior ($q_\phi \approx p(\mathbf{z})$, KL $\to 0$). This gives low ELBO loss but completely uninformative latents. Causes and remedies:

- *Cause:* overpowered decoder (e.g., autoregressive decoder) can model $p(\mathbf{x})$ without $\mathbf{z}$.

- *KL annealing:* start training with $\beta = 0$ (pure reconstruction), linearly increase to $\beta = 1$ over the first T_{warm} steps.
- *Free bits:* only penalise KL if $\mathrm{KL}(q_i \| p) > \lambda$ per dimension; allow "free information" below threshold λ.
- *Constrained optimisation:* target a specific KL value C via Lagrange multiplier: min $|\mathrm{KL} - C|$ (controlled capacity increase).

Example — VAE Training on CelebA

Architecture: encoder ResNet-18 $\to (\square, \log \square^2) \in \mathbb{R}^{2\times 128}$; decoder MLP upsampler $\to 64 \times 64 \times 3$.
Typical training trajectory:

Epoch	Recon loss	KL	ELBO
1	1850	0.3	1850
10	620	14.2	634
50	285	32.1	317
100	240	38.4	278

KL increases as the encoder learns informative posteriors; reconstruction improves as the decoder learns to decode from informative codes. Balance is reached around epoch 50–100.

18.1 The Change-of-Variables Formula

Change-of-variables formula. Let $\mathbf{z}_0 \sim p_0(\mathbf{z}_0)$ be a base distribution and $f : \mathbb{R}^d \to \mathbb{R}^d$ a diffeomorphism (differentiable bijection with differentiable inverse). Then $\mathbf{z}_1 = f(\mathbf{z}_0)$ has density:

$$p_1(\mathbf{z}_1) = p_0(f^{-1}(\mathbf{z}_1)) \cdot \left|\det\left(\frac{\partial f^{-1}}{\partial \mathbf{z}_1}\right)\right| = p_0(f^{-1}(\mathbf{z}_1)) \cdot \left|\det\left(\frac{\partial f}{\partial \mathbf{z}_0}\right)\right|^{-1}$$

Normalising Flow: definition

A *normalising flow* is a composition of K invertible transformations:

$$\mathbf{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0), \qquad \mathbf{z}_0 \sim p_0$$

The log-density at the output is:

$$\log p_K(\mathbf{z}_K) = \log p_0(\mathbf{z}_0) - \sum_{k=1}^{K} \log \left|\det\left(\frac{\partial f_k}{\partial \mathbf{z}_{k-1}}\right)\right|$$

Computational cost. The Jacobian determinant of a $d \times d$ matrix costs $O(d^3)$ in general. For high-dimensional models ($d = 1024$ for 32×32 images) this is prohibitive. Normalising flows are designed with special Jacobian structures that make computation $O(d)$ or $O(d \log d)$:

- **Triangular flows** (autoregressive flows, RealNVP): triangular Jacobian, $\det = \prod_i \partial_i$.
- **Sylvester flows**: rank-r perturbation of the identity, det via the matrix determinant lemma.

- **Continuous flows** (CNF): Jacobian trace via the continuity equation (Section 21.1).

Training a normalising flow. Maximise the log-likelihood on the training set:

$$\max_\theta \frac{1}{N}\sum_{i=1}^{N} \log p_K(\mathbf{x}_i;\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[\log p_0(f_\theta^{-1}(\mathbf{x}_i)) - \sum_k \log\left|\det J_{f_k}\right|\right]$$

There are no latent variables to infer: the model is *exactly invertible*. Sampling: $\mathbf{z}_0 \sim p_0$, then $\mathbf{x} = f_K \circ \cdots \circ f_1(\mathbf{z}_0)$.

Example — RealNVP: Affine Coupling Layer

Partition $\mathbf{z} = (z_{1:m}, z_{m+1:d})$. The coupling layer is:

$$z'_{1:m} = z_{1:m}, \qquad z'_{m+1:d} = z_{m+1:d} \odot \exp(s_\theta(z_{1:m})) + t_\theta(z_{1:m})$$

where $s_\theta, t_\theta : \mathbb{R}^m \to \mathbb{R}^{d-m}$ are arbitrary neural networks. The Jacobian is block-triangular with $\det = \exp(\sum_j s_{\theta,j}(z_{1:m}))$. Inversion: $z_{m+1:d} = (z'_{m+1:d} - t_\theta(z_{1:m})) \odot \exp(-s_\theta(z_{1:m}))$.

Theorem: Change of Variables

Let $\mathbf{z} \sim p(\mathbf{z})$ and $\mathbf{x} = f(\mathbf{z})$ where $f : \mathbb{R}^d \to \mathbb{R}^d$ is differentiable and invertible. Then:

$$p(\mathbf{x}) = p(\mathbf{z}) \cdot \left|\det J_f(\mathbf{z})\right|^{-1} = p(f^{-1}(\mathbf{x})) \cdot \left|\det J_{f^{-1}}(\mathbf{x})\right|$$

where J_f is the Jacobian matrix of f: $(J_f)_{ij} = \dfrac{\partial f_i}{\partial z_j}$.

Intuition. A small volume element $d\mathbf{z}$ around $\mathbf{z}$ maps to volume element $|\det J_f|\, d\mathbf{z}$ around $\mathbf{x} = f(\mathbf{z})$. For probability mass to be conserved, the density must scale inversely.

Log probability:

$$\log p(\mathbf{x}) = \log p(\mathbf{z}) - \log\left|\det J_f(\mathbf{z})\right|$$

Composing flows. For a chain $\mathbf{z}_0 \xrightarrow{f_1} \mathbf{z}_1 \xrightarrow{f_2} \cdots \xrightarrow{f_K} \mathbf{z}_K = \mathbf{x}$:

$$\log p(\mathbf{x}) = \log p_0(\mathbf{z}_0) - \sum_{k=1}^{K} \log\left|\det J_{f_k}(\mathbf{z}_{k-1})\right|$$

The design challenge: choose f_k such that $\det J_{f_k}$ is cheap to compute (ideally $O(d)$).

Invertibility requirement. The change-of-variables formula $p_X(\mathbf{x}) = p_Z(f^{-1}(\mathbf{x}))|\det J_{f^{-1}}(\mathbf{x})|$ requires f to be a diffeomorphism (smooth bijection with smooth inverse). The key computational challenge: evaluating $|\det J_f(\mathbf{z})|$ efficiently.

Why general neural networks fail. A fully-connected layer $\mathbf{x} = \mathbf{W}\mathbf{z} + \mathbf{b}$ has $|\det \mathbf{W}|$ as its Jacobian determinant. $O(d^3)$ cost for exact computation; $O(d^2)$ storage. For $d = 256 \times 256 \times 3 \approx 200,000$ (images), completely intractable. Normalising flows must use architectures with cheap Jacobian determinants.

Architectures with $O(d)$ log-determinant.

Architecture	Log-det cost	Example
Diagonal $\mathbf{W}$	$O(d)$	Affine normalisation
Triangular Jacobian	$O(d)$	Coupling layers (RealNVP)
1×1 conv (LU)	$O(d)$	Glow
Autoregressive	$O(d)$ parallel	MAF, IAF
Continuous (ODE)	$O(d)$ Hutchinson	CNFs

Maximum likelihood training. Given a dataset $\{\mathbf{x}_n\}$, the training objective is:

$$\max_\theta \frac{1}{N}\sum_n \log p_\theta(\mathbf{x}_n) = \frac{1}{N}\sum_n \left[\log p_Z(f_\theta^{-1}(\mathbf{x}_n)) + \log\left|\det J_{f_\theta^{-1}}(\mathbf{x}_n)\right|\right]$$

No approximation: this is the exact log-likelihood. Backprop through the log-Jacobian term requires differentiating through the determinant computation.

18.2 Autoregressive Flows

Definition: Autoregressive Flow

An autoregressive flow applies the transform:

$$x_i = f(z_i;\, h_i), \qquad h_i = g(z_1, \dots, z_{i-1})$$

where $f(\cdot\,; h_i)$ is monotone (ensuring invertibility) and g is any neural network.

Jacobian is triangular. Since x_i depends only on $z_1, \dots, z_i$:

$$J = \begin{pmatrix} \frac{\partial x_1}{\partial z_1} & 0 & \cdots & 0 \\ \frac{\partial x_2}{\partial z_1} & \frac{\partial x_2}{\partial z_2} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ \frac{\partial x_d}{\partial z_1} & \cdots & & \frac{\partial x_d}{\partial z_d} \end{pmatrix}$$

$\det J = \prod_{i=1}^d \frac{\partial x_i}{\partial z_i}$: only diagonal elements, cost $O(d)$.

Affine autoregressive flow (MAF/IAF). Choose $f(z_i; h_i) = \mu(h_i) + \sigma(h_i) \cdot z_i$:

$$x_i = \mu_i + \sigma_i z_i, \quad \frac{\partial x_i}{\partial z_i} = \sigma_i$$

$$\log|\det J| = \sum_{i=1}^d \log|\sigma_i|$$

Masked Autoencoder for Flows (MADE). The context $h_i = g(z_1, \dots, z_{i-1})$ can be computed for all i simultaneously using a single masked neural network, where the mask ensures causal dependence.

Example — Affine flow in $d = 3$

Let $\mathbf{z} = (0.5, -0.3, 1.2)$, computed parameters $\square = (0.1, 0.2, -0.1)$, $\square = (0.8, 1.2, 0.9)$. Forward ($\mathbf{z} \to \mathbf{x}$):

$$x_1 = 0.1 + 0.8 \cdot 0.5 = 0.500 \quad x_2 = 0.2 + 1.2 \cdot (-0.3) = -0.160 \quad x_3 = -0.1 + 0.9 \cdot 1.2 = 0.980$$

Log-Jacobian: $\log|0.8| + \log|1.2| + \log|0.9| = -0.2231 + 0.1823 - 0.1054 = -0.1462$
$p(\mathbf{x})$: add $-(-0.1462) = +0.1462$ to $\log p(\mathbf{z})$ — the flow has spread the distribution.

Masked Autoregressive Flow (MAF). MAF (Papamakarios et al., 2017) uses a MADE (Masked Autoencoder for Distribution Estimation) network to implement the autoregressive conditionals:

$$s_i = h_\theta^{(s)}(x_{1:i-1}), \quad t_i = h_\theta^{(t)}(x_{1:i-1}), \quad z_i = (x_i - t_i)\exp(-s_i)$$

Masking ensures the MADE output for dimension i depends only on $x_{1:i-1}$ — the autoregressive structure is enforced by zeroing out the weight matrix entries that would create forward dependencies.

Inverse Autoregressive Flow (IAF). IAF (Kingma et al., 2016) reverses the direction: given $\mathbf{z}$, compute $\mathbf{x}$ autoregressively. The sampling $\mathbf{x} \sim p_\theta$ is fast (d sequential steps with MADE); the density evaluation $p_\theta(\mathbf{x})$ requires d sequential steps. MAF and IAF are inverses of each other in terms of speed tradeoffs:

Method	Density eval	Sample generation
MAF	$O(d)$ parallel	$O(d)$ sequential
IAF	$O(d)$ sequential	$O(d)$ parallel
Coupling (RealNVP)	$O(1)$	$O(1)$

Parallel Wavenet. Van den Oord et al. (2018) train a MAF teacher and then distil it into an IAF student via probability density distillation: the IAF student learns to match the MAF teacher's distribution while enabling fast parallel synthesis. This enables real-time speech synthesis (Parallel WaveNet runs at $500\times$ real-time on CPU).

Universality of autoregressive flows. Any continuous density p on $\mathbb{R}^d$ can be approximated arbitrarily well by a MAF or IAF with sufficiently expressive conditioners g_i. Proof sketch: by the probability integral transform, $F_i(x_i|\mathbf{x}_{<i}) \sim \mathcal{U}[0,1]$, so the composition $z_i = F_i(x_i|\mathbf{x}_{<i})$ is a valid normalising flow from p to $\mathcal{U}[0,1]^d$. Composing with an elementwise logit gives a flow to $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Thus MAFs are universal density models (given exact CDFs as conditioners) — practical approximation quality depends on the expressiveness of the neural network conditioners.

Normalising flows for variational inference. One important application of IAFs is improving the variational posterior in the VAE. Instead of $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \square_\phi(\mathbf{x}), \mathrm{diag}(\square_\phi^2(\mathbf{x})))$, apply K IAF steps:

$$\mathbf{z}_0 \sim \mathcal{N}(\square_\phi, \mathrm{diag}(\square_\phi^2)), \quad \mathbf{z}_k = f_k(\mathbf{z}_{k-1}; \mathbf{h}_\phi(\mathbf{x}))$$

where $\mathbf{h}_\phi(\mathbf{x})$ is a context vector from the encoder used to condition each flow step. The ELBO becomes:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_\square\left[\log p_\theta(\mathbf{x}|\mathbf{z}_K) + \log p(\mathbf{z}_K) - \log q_0(\mathbf{z}_0) + \sum_{k=1}^{K} \log|\det J_{f_k}|\right]$$

Because IAFs are parallel at inference, the forward pass through K flow steps is $O(Kd)$ — comparable to K linear layers. This variant (IAF-VAE, Kingma et al., 2016) achieves significantly lower ELBO gap than mean-field VAEs on binarised MNIST.

18.3 Coupling Layers and RealNVP

Coupling layers (Dinh et al. 2017) are a practical implementation of normalising flows:

Split $\mathbf{z} \in \mathbb{R}^d$ into $(\mathbf{z}_{1:k}, \mathbf{z}_{k+1:d})$. The coupling transformation:

$$\mathbf{x}_{1:k} = \mathbf{z}_{1:k}, \qquad \mathbf{x}_{k+1:d} = \mathbf{z}_{k+1:d} \odot \exp(s(\mathbf{z}_{1:k})) + t(\mathbf{z}_{1:k})$$

where s and t are arbitrary neural networks.

Key properties.

- **Triangular Jacobian**: $\partial \mathbf{x}/\partial \mathbf{z}$ is lower triangular with diagonal entries $\exp(s_j(\mathbf{z}_{1:k}))$. Determinant: $\prod_j \exp(s_j) = \exp(\sum_j s_j)$ — computed in $O(d)$.
- **Invertible**: $\mathbf{z}_{k+1:d} = (\mathbf{x}_{k+1:d} - t(\mathbf{x}_{1:k})) \odot \exp(-s(\mathbf{x}_{1:k}))$.
- **Parallelisable**: forward and inverse pass are both $O(d)$ per example.

Stacking alternating splits (odd/even dimensions) creates expressive bidirectional flows (RealNVP architecture).

Multi-scale architecture (RealNVP). To improve expressiveness, RealNVP alternates the partition (which variables are "passed through" unchanged) using a checkerboard or channel-wise split, and applies a *squeezing* operation to trade spatial resolution for channels. Between coupling blocks, half the variables are factored out:

$$\log p(\mathbf{z}) = \log p(\mathbf{z}_1) + \log p(\mathbf{z}_2|\mathbf{z}_1) = \sum_{\text{layers}} \log |J_l|$$

where each $\mathbf{z}_1$ factored at an intermediate layer is passed through a standard Gaussian without further transformations.

Practical implementation. s and t are typically deep ResNets. Batch normalisation is applied between coupling layers (with a learned shift/scale to remain invertible). For images:

- Input: $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$.
- Checkerboard partition: alternate even/odd spatial positions.
- Channel partition: first/second half of channels.
- Squeeze: $2 \times 2 \times C \to 1 \times 1 \times 4C$ (folds spatial into channels).

Example — Log-Likelihood of RealNVP

A 4-coupling-layer model with Gaussian base. For input $\mathbf{x} \in \mathbb{R}^{64}$:

$$\log p(\mathbf{x}) = \log p_Z(\mathbf{z}) + \sum_{l=1}^{4} \sum_j s_j^{(l)}(\mathbf{z}_{1:k}^{(l)})$$

where $\mathbf{z} = f^{-1}(\mathbf{x})$ and the sum of log-scale outputs s_j is the log-Jacobian contribution. The model is trained by maximising this exactly (no approximation, unlike VAE).

RealNVP architecture details. Each coupling layer in RealNVP partitions $\mathbf{z} = (\mathbf{z}_{1:d/2}, \mathbf{z}_{d/2+1:d})$. The transformation is:

$$\mathbf{y}_{1:d/2} = \mathbf{z}_{1:d/2}$$

$$\mathbf{y}_{d/2+1:d} = \mathbf{z}_{d/2+1:d} \odot \exp(s(\mathbf{z}_{1:d/2})) + t(\mathbf{z}_{1:d/2})$$

where s and t are arbitrary neural networks (scale and translation). The Jacobian is triangular:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{z}} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \frac{\partial \mathbf{y}_{d/2+1:d}}{\partial \mathbf{z}_{1:d/2}} & \operatorname{diag}(\exp(s(\mathbf{z}_{1:d/2}))) \end{pmatrix}$$

The log-determinant is $\sum_i s_i(\mathbf{z}_{1:d/2})$ — a sum over $d/2$ scalar outputs of s, computed in $O(d)$ regardless of the complexity of s. The inverse is equally simple:

$$\mathbf{z}_{d/2+1:d} = (\mathbf{y}_{d/2+1:d} - t(\mathbf{y}_{1:d/2})) \odot \exp(-s(\mathbf{y}_{1:d/2}))$$

Masking strategies. To ensure all dimensions are transformed (not just the second half), alternating coupling layers use different partitions:

- *Spatial checkerboard masking* (for images): odd-positioned pixels are "active" in odd layers, even-positioned in even layers.
- *Channel-wise masking*: for feature maps, alternate which channels are passed through s, t and which are transformed.
- *Random permutation*: randomly permute dimensions between layers. Used in GLOW.

Multi-scale architecture. RealNVP factorises the density at multiple resolutions. After every 2 coupling layers, half the channels are split off and modelled directly as Gaussian:

$$p(\mathbf{z}) = \prod_{\ell=1}^{L} \mathcal{N}(\mathbf{h}_\ell; \square_\ell, \operatorname{diag}(\square_\ell^2))$$

where $\mathbf{h}_\ell$ are the split-off channels at level ℓ. This early commitment to simpler structure at lower levels speeds training, reduces memory, and allows efficient sampling at multiple resolutions.

Example — Computing the log-likelihood in RealNVP

For image $\mathbf{x} \in [0,1]^{H \times W \times C}$, the log-likelihood is:

1. Dequantize: $\mathbf{x}' = \mathbf{x} + \mathcal{U}(0, 1/256)$, then logit-transform: $\mathbf{z}^{(0)} = \operatorname{logit}(\mathbf{x}')$.
2. Apply L coupling layers: $\mathbf{z}^{(0)} \xrightarrow{f_1} \mathbf{z}^{(1)} \xrightarrow{f_2} \cdots \xrightarrow{f_L} \mathbf{z}^{(L)}$.
3. Sum log-determinants: $\sum_\ell \sum_i s_i^{(\ell)}$.
4. Evaluate base density: $\log \mathcal{N}(\mathbf{z}^{(L)}; \mathbf{0}, \mathbf{I})$.
5. Total: $\log p(\mathbf{x}) = \log p(\mathbf{z}^{(L)}) + \sum_\ell \log |\det J_\ell| - \log |\det J_{\text{logit}}|$.

Bits-per-dimension: $\text{bpd} = -\log_2 p(\mathbf{x}) / (H \cdot W \cdot C)$. RealNVP achieves ~ 3.49 bpd on CIFAR-10.

Example — Computing the log-likelihood in RealNVP

For image $\mathbf{x} \in [0,1]^{H \times W \times C}$, the log-likelihood is:

1. Dequantize: $\mathbf{x}' = \mathbf{x} + \mathcal{U}(0, 1/256)$, then logit-transform: $\mathbf{z}^{(0)} = \operatorname{logit}(\mathbf{x}')$.
2. Apply L coupling layers: $\mathbf{z}^{(0)} \xrightarrow{f_1} \mathbf{z}^{(1)} \xrightarrow{f_2} \cdots \xrightarrow{f_L} \mathbf{z}^{(L)}$.
3. Sum log-determinants: $\sum_\ell \sum_i s_i^{(\ell)}$.

4. Evaluate base density: $\log \mathcal{N}(\mathbf{z}^{(L)}; \mathbf{0}, \mathbf{I})$.
5. Total: $\log p(\mathbf{x}) = \log p(\mathbf{z}^{(L)}) + \sum_\ell \log |\det J_\ell| - \log |\det J_{\text{logit}}|$.

Bits-per-dimension: $\text{bpd} = -\log_2 p(\mathbf{x}) / (H \cdot W \cdot C)$. RealNVP achieves ~ 3.49 bpd on CIFAR-10.

18.4 Glow: Generative Flow with Invertible 1x1 Convolutions

Glow (Kingma & Dhariwal 2018) adds an invertible 1×1 convolution between coupling layers. This is a learned permutation of channels, generalising the channel splitting in RealNVP.

For a 1×1 convolution $\mathbf{W} \in \mathbb{R}^{C \times C}$:

$$\log |\det(\mathbf{W})| = \log |\det \mathbf{W}|$$

Computed cheaply via LU decomposition: $\mathbf{W} = \mathbf{PLU}$ where $\mathbf{P}$ is a permutation, $\mathbf{L}$ lower-triangular with unit diagonal, $\mathbf{U}$ upper-triangular. Then:

$$\log |\det \mathbf{W}| = \sum_i \log |U_{ii}|$$

PART 19

Generative Adversarial Networks (GANs)

Two networks compete: a generator produces fake samples, a discriminator distinguishes real from fake. We derive the minimax objective, its optimal solution, and the Wasserstein distance formulation.

Glow. Kingma & Dhariwal (2018) combine: (1) squeeze (spatial reshape), (2) ActNorm (activation normalisation), (3) invertible 1×1 convolutions (learned permutation), (4) affine coupling layers. Result: high-quality 256×256 image generation with exact density.

ActNorm. Glow replaces BatchNorm with *activation normalisation*: a per-channel affine transform $\mathbf{h} = \mathbf{s} \odot \mathbf{x} + \mathbf{b}$ with learned $\mathbf{s}, \mathbf{b} \in \mathbb{R}^C$. Initialised so that the first mini-batch has zero mean and unit variance per channel. This is data-dependent initialisation rather than running statistics — avoids batch size dependence.

Glow step. Each step in Glow consists of three operations:

1. **ActNorm**: $\mathbf{h} = \mathbf{s} \odot \mathbf{x} + \mathbf{b}$, $\log|\det J| = H \cdot W \cdot \sum_c \log|s_c|$.
2. **Invertible 1×1 conv**: $\mathbf{h} = \mathbf{W}\mathbf{z}$, $\log|\det J| = H \cdot W \cdot \log|\det \mathbf{W}|$.
3. **Affine coupling**: $\mathbf{h} = [\mathbf{x}_1;\ \mathbf{x}_2 \odot \exp(s(\mathbf{x}_1)) + t(\mathbf{x}_1)]$.

Total log-determinant is the sum across all steps, computed in $O(HWC)$ per step.

Practical capabilities. Glow trained on CelebA-HQ (256×256) enables:

- High-quality image synthesis by sampling $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and decoding.
- Semantic manipulation: edit face attributes (smile, age) by moving $\mathbf{z}$ in the direction learned by a linear classifier on attribute labels.
- Exact likelihood: $\log p(\mathbf{x})$ is computable in one forward pass (unlike VAE which gives a lower bound, unlike GANs which give no likelihood).

Limitation vs. diffusion models. Glow's expressiveness is limited by the affine coupling structure: each coupling layer can only transform half the dimensions. Diffusion models achieve higher sample quality (FID ≈1–3 for DDPM on CIFAR-10 vs. ≈30 for Glow) at the cost of slower sampling.

1×1 convolution as a learned permutation. In Glow, the fixed channel permutation between coupling layers is replaced by a learned 1×1 convolution with weight matrix $\mathbf{W} \in \mathbb{R}^{C\times C}$. For a feature map $\mathbf{h} \in \mathbb{R}^{H\times W\times C}$, the transformation applies $\mathbf{W}$ to each spatial location:

$$\mathbf{y}_{ij} = \mathbf{W}\mathbf{h}_{ij}$$

The log-determinant of the Jacobian (for all HW spatial locations) is:

$$\log|\det J| = HW \cdot \log|\det \mathbf{W}|$$

$\mathbf{W}$ is initialised as a random rotation (via QR decomposition), so $|\det \mathbf{W}| = 1$ initially. The PLU decomposition $\mathbf{W} = \mathbf{PLU}$ with fixed permutation $\mathbf{P}$, lower triangular $\mathbf{L}$, and upper triangular $\mathbf{U}$ reduces cost from $O(C^3)$ to $O(C^2)$ for the determinant: $\log|\det \mathbf{W}| = \sum_i \log|U_{ii}|$.

ActNorm. To stabilise training with large batches, Glow introduces *ActNorm*: a learnable per-channel affine transformation $\mathbf{y} = s \odot \mathbf{x} + \mathbf{b}$ where $\mathbf{s}, \mathbf{b} \in \mathbb{R}^C$ are initialised so that the first mini-batch has zero mean and unit variance (data-dependent initialisation). The log-determinant contribution is $HW \cdot \sum_i \log|s_i|$. ActNorm replaces batch normalisation (which is not invertible) in the flow.

Glow architecture summary. Each *step of flow* consists of: ActNorm → 1×1 convolution → affine coupling layer. Steps are stacked K times at each resolution level. After every level, the spatial resolution is halved (squeeze operation) and half the channels are factored out. With L levels and K steps per level, total parameters scale as $O(LKC^2)$. Glow with $L = 3, K = 32$ on 256×256 images achieves 3.35 bpd on ImageNet, generating photo-realistic 256×256 faces with semantically meaningful latent space interpolations.

Flow++. Ho et al. (2019) improve on Glow by replacing affine coupling (restricted to scale-and-shift) with *logistic mixture CDFs* as coupling functions:

$$y_i = \text{CDF}_{\text{mix}}(z_i;\, \pi(\mathbf{z}_{1:d/2}),\, \mu(\mathbf{z}_{1:d/2}),\, s(\mathbf{z}_{1:d/2}))$$

where the CDF of a mixture of K logistics is monotone and invertible. The log-derivative (Jacobian contribution) is $\log \text{pdf}_{\text{mix}}(z_i; \ldots)$, computable in $O(K)$. This more expressive coupling reduces bpd from 3.69 (Glow on CIFAR-10) to 3.08, matching some autoregressive models while maintaining $O(d)$ parallel sampling.

Comparison of normalising flow architectures.

Model	Coupling	Jacobian cost	Sample speed	CIFAR-10 bpd
NICE	Additive	$O(d)$	$O(d)$	4.48
RealNVP	Affine	$O(d)$	$O(d)$	3.49
MAF	Autoregressive	$O(d)$	$O(d^2)$	3.08
IAF	Inverse autoregressive	$O(d)$	$O(d)$	$\sim$3.2
Glow	Affine + 1×1 conv	$O(d)$	$O(d)$	3.35
Flow++	Logistic mixture CDF	$O(Kd)$	$O(d)$	3.08

Normalising flows vs. diffusion models (2024 perspective). Flows provide *exact* likelihood and *single-pass* sampling (one forward pass through all layers). Diffusion models require iterative sampling (10–1000 passes). However, diffusion models achieve FID $\approx$ 1–3 on CIFAR-10 while Glow achieves FID ≈ 46. The quality gap is because affine coupling layers have limited expressiveness per step; diffusion models effectively compose thousands of simple denoisers. Flow matching (§21.2–21.3) bridges this gap by training a flow with the efficiency of coupling layers but the flexibility of diffusion trajectories.

19.1 The Minimax Objective

Generative Adversarial Networks (GAN, Goodfellow et al., 2014) formulate generation as a zero-sum game between two networks: a generator G that produces synthetic samples, and a discriminator D that tries to distinguish real from fake samples.

GAN Minimax Objective

$$\min_G \max_D \; V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z}[\log(1 - D(G(\mathbf{z})))]$$

- $G : \mathcal{Z} \to \mathcal{X}$: generator, maps noise $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to fake samples.
- $D : \mathcal{X} \to [0, 1]$: discriminator, estimates $P(\text{real}|\mathbf{x})$.

Optimal discriminator. For fixed G, the optimal discriminator is:

$$D^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}$$

The optimal discriminator returns the Bayesian probability that $\mathbf{x}$ is real.

Connection with JSD. Substituting D^* into the objective:

$$V(D^*, G) = -\log 4 + 2\, D_{\text{JSD}}(p_{\text{data}} \| p_G)$$

where $D_{\text{JSD}}(P\|Q) = \frac{1}{2} D_{\text{KL}}(P\|\frac{P+Q}{2}) + \frac{1}{2} D_{\text{KL}}(Q\|\frac{P+Q}{2})$ is the Jensen-Shannon Divergence. The global minimum $V(D^*, G^*) = -\log 4$ is reached when $p_G = p_{\text{data}}$.

Training instability: the gradient problem. If D is too strong, $D(G(\mathbf{z})) \approx 0$ for all fake samples. The gradient $\partial \log(1 - D(G(\mathbf{z})))/\partial G \approx 0$: the generator receives no useful information. Heuristic fix: minimise $-\log D(G(\mathbf{z}))$ instead of $\log(1 - D(G(\mathbf{z})))$ (non-saturating loss).

Example — Step-by-Step Update

Discriminator step (parameters ϕ):

$$\nabla_\phi \left[\frac{1}{B} \sum_i \log D_\phi(\mathbf{x}_i) + \frac{1}{B} \sum_j \log(1 - D_\phi(G_\theta(\mathbf{z}_j))) \right]$$

Gradient ascent (maximisation).
Generator step (parameters θ):

$$-\nabla_\theta \frac{1}{B} \sum_j \log D_\phi(G_\theta(\mathbf{z}_j))$$

Gradient descent (minimisation, non-saturating).

Practice. A typical cycle: $n_D = 1$ or 5 discriminator steps per generator step. Tricks: gradient penalty (WGAN-GP), spectral normalisation of the discriminator, progressive growing (ProGAN), self-attention (SAGAN).

Alternating training. In practice, the minimax GAN is trained with alternating updates:

1. For k steps: update D_ϕ maximising $\mathbb{E}[\log D(\mathbf{x})] + \mathbb{E}[\log(1 - D(G(\mathbf{z})))]$.
2. For 1 step: update G_θ minimising $\mathbb{E}[\log(1 - D(G(\mathbf{z})))]$ or equivalently maximising $\mathbb{E}[\log D(G(\mathbf{z}))]$ (non-saturating loss).

Typically $k = 1$ (same number of steps for D and G); in WGAN $k = 5$ is used to bring the critic close to its optimum.

Mode collapse: mechanism and diagnosis. The generator may converge to producing few modes (mode collapse): the discriminator learns to reject them, the generator switches to other modes, producing cyclic oscillation without convergence. Diagnosis: monitor diversity (number of modes in a 50K sample set), FID, and the distribution of discriminator outputs.

Conditional GAN (cGAN). Mirza & Osindero (2014): provide both generator and discriminator with a label $\mathbf{y}$ (class, text embedding):

$$\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\log D(\mathbf{x}, \mathbf{y})] + \mathbb{E}_{\mathbf{z},\mathbf{y}}[\log(1 - D(G(\mathbf{z}, \mathbf{y}), \mathbf{y}))]$$

Enables conditional generation. BigGAN (Brock et al., 2019) scales cGANs to ImageNet with batches of 2048 and spectral regularisation techniques.

Optimal discriminator and generator loss. For a fixed generator G, the optimal discriminator is:

$$D^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}$$

Proof. Maximise $\int [p_{\text{data}} \log D + p_G \log(1-D)]\, d\mathbf{x}$ pointwise: $\frac{d}{dD}[a \log D + b \log(1-D)] = a/D - b/(1-D) = 0 \Rightarrow D = a/(a+b)$.

Substituting D^* into the minimax objective:

$$\min_G V(G, D^*) = -\log 4 + 2\,\text{JSD}(p_{\text{data}} \| p_G)$$

Since $\text{JSD} \geq 0$, the global optimum is $\text{JSD} = 0 \Leftrightarrow p_G = p_{\text{data}}$ with minimum value $-\log 4$.

Training instability from JSD. When p_{data} and p_G have disjoint supports (common at the start of training), $\text{JSD}(p_{\text{data}} \| p_G) = \log 2$ (constant) and $\nabla_\theta \text{JSD} = 0$ — the generator gradient vanishes. This is why GAN training requires carefully balanced updates and motivates WGAN (§19.2).

Practical training tricks.

- *k discriminator steps per generator step:* train D more than G to keep D near-optimal. Typical: $k = 1$–5.
- *Label smoothing:* use 0.9 instead of 1.0 for real labels in D's loss — reduces overconfidence and gradient vanishing.
- *Instance noise:* add small Gaussian noise $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ to both real and fake samples, decreasing σ over training. Smooths the distributions, preventing disjoint supports.
- *Two-timescale update rule (TTUR):* use different learning rates for G and D, convergence guaranteed under certain conditions (Heusel et al., 2017).

19.2 Wasserstein GAN

The problem with classical metrics. The KL divergence and the Jensen-Shannon divergence (JSD) between distributions p_r (real) and p_g (generative) can be infinite or constant when their supports do not overlap — a typical situation at the start of training. This causes zero gradients for the generator.

Wasserstein-1 distance (Earth Mover's Distance).

$$W(p_r, p_g) = \inf_{\gamma \in \Pi(p_r, p_g)} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma}[\|\mathbf{x} - \mathbf{y}\|]$$

where $\Pi(p_r, p_g)$ is the set of all couplings with marginals p_r and p_g. This metric is finite and provides informative gradients even for distributions with disjoint support.

Kantorovich–Rubinstein Duality

For distributions on a compact metric space:

$$W(p_r, p_g) = \sup_{\|f\|_L \leq 1} \left\{ \mathbb{E}_{\mathbf{x}\sim p_r}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_g}[f(\mathbf{x})] \right\}$$

where the supremum is taken over 1-Lipschitz functions $f : \mathbb{R}^d \to \mathbb{R}$ ($|f(\mathbf{x}) - f(\mathbf{y})| \leq \|\mathbf{x} - \mathbf{y}\|$).

WGAN-GP (Gradient Penalty). Gulrajani et al. (2017) propose penalising the critic gradient:

$$\mathcal{L}_{\text{GP}} = \lambda\, \mathbb{E}_{\hat{\mathbf{x}}\sim\hat{p}} \left[\left(\|\nabla_{\hat{\mathbf{x}}} D_w(\hat{\mathbf{x}})\|_2 - 1 \right)^2 \right]$$

where $\hat{\mathbf{x}} = t\mathbf{x} + (1-t)\tilde{\mathbf{x}}$ with $t \sim \text{Uniform}(0,1)$, $\mathbf{x} \sim p_r$, $\tilde{\mathbf{x}} \sim p_g$. The total critic loss becomes:

$$\mathcal{L}_D = \mathbb{E}_{p_g}[D_w] - \mathbb{E}_{p_r}[D_w] + \lambda\, \mathbb{E}_{\hat{p}}[(\|\nabla D_w\|_2 - 1)^2]$$

Example — GAN vs. WGAN-GP: stability

The classical GAN suffers from:

- **Mode collapse**: the generator produces few modes of the target distribution.
- **Oscillations**: discriminator/generator do not converge to an equilibrium.
- **Vanishing gradient**: a too-strong discriminator $\Rightarrow$ gradient ≈ 0 for the generator.

WGAN-GP mitigates these problems: the Wasserstein distance provides gradient signal even when the distributions do not overlap, and the gradient penalty stabilises the critic's training.

Theoretical convergence. If the critic is optimised to convergence at each step, the generator gradient is:

$$\nabla_\theta W(p_r, p_\theta) = -\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\nabla_\theta D_w^*(G_\theta(\mathbf{z}))]$$

Under regularity conditions, the generator gradient flow converges to the real distribution p_r.

Problem with vanilla GAN. When supports of p_{data} and p_θ do not overlap, JSD $= \log 2$ constantly — no gradient signal for G.

Wasserstein-1 Distance

$$W(p,q) = \inf_{\gamma \in \Pi(p,q)} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma}[\|\mathbf{x}-\mathbf{y}\|]$$

where $\Pi(p,q)$ is the set of all joint distributions with marginals p and q. Informally: the minimum "earth-moving cost" to transform p into q.

Kantorovich-Rubinstein duality. By the Kantorovich duality theorem:

$$W(p,q) = \sup_{\|f\|_L \le 1} \left(\mathbb{E}_p[f(\mathbf{x})] - \mathbb{E}_q[f(\mathbf{y})]\right)$$

where the supremum is over all 1-Lipschitz functions f.

WGAN objective. Replace the discriminator with a 1-Lipschitz *critic* f_ϕ:

$$\min_\theta \max_{\phi:\|f_\phi\|_L \le 1} \mathbb{E}_{p_{\text{data}}}[f_\phi(\mathbf{x})] - \mathbb{E}_{p_z}[f_\phi(G_\theta(\mathbf{z}))]$$

Enforcing the Lipschitz constraint: weight clipping (original WGAN), or gradient penalty (WGAN-GP):

$$\mathcal{L}_{\text{GP}} = \lambda\, \mathbb{E}_{\hat{\mathbf{x}}}\left[\left(\left\|\nabla_{\hat{\mathbf{x}}} f_\phi(\hat{\mathbf{x}})\right\|_2 - 1\right)^2\right]$$

where $\hat{\mathbf{x}}$ is sampled uniformly along lines between real and fake samples.

19.3 Training Dynamics and Mode Collapse

Equilibrium analysis. At Nash equilibrium, the discriminator outputs $D^*(\mathbf{x}) = p_{\text{data}}(\mathbf{x})/(p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x}))$. The generator's optimal loss at this D^* equals $-\log 4 + 2\,\text{JSD}(p_{\text{data}} \,\|\, p_G)$ where JSD is the Jensen-Shannon divergence.

Mode collapse. The generator learns to produce a few high-quality samples that fool the discriminator, ignoring most of the data distribution. Symptoms: generated samples all look similar; FID/IS scores poor.

Mitigation strategies.

- Minibatch discrimination: compare generated samples to each other.
- Spectral normalisation: control Lipschitz constant of discriminator.
- Gradient penalty (WGAN-GP): ensure discriminator satisfies 1-Lipschitz constraint.

Quantitative diagnosis. Let p_G be the generator distribution and p_{data} the data distribution. Mode collapse manifests as low support: $\text{supp}(p_G) \subset \text{supp}(p_{\text{data}})$ with $\text{Var}_{p_G}[\mathbf{x}] \ll \text{Var}_{p_{\text{data}}}[\mathbf{x}]$.

Training instability: the vanishing gradient problem. When D is near-optimal (accurate), $\log(1 - D(\mathbf{x}))$ saturates for generated $\mathbf{x}$: $D(G(\mathbf{z})) \to 0$, so $\log(1 - D) \to 0$ and the generator gradient $\nabla_\theta \log(1 - D(G(\mathbf{z}))) \to 0$. The non-saturating objective $-\log D(G(\mathbf{z}))$ (used in practice) mitigates this.

Wasserstein distance as remedy. WGAN (§19.2) replaces the JSD with $W_1(p_{\text{data}}, p_G)$, which provides a useful gradient even when the supports are disjoint (JSD is flat in that case). This is the main theoretical motivation for WGAN.

Mitigation strategies (extended).

- *Minibatch discrimination:* compute statistics across the batch (e.g. pairwise ℓ_2 distances between samples). The discriminator receives these statistics alongside each sample, making it harder to fool with identical outputs.
- *Spectral normalisation* (§19.4): constrains $\|\mathbf{W}\|_2 \leq 1$ at each layer, stabilising training without WGAN's Lipschitz penalty.
- *Gradient penalty* (WGAN-GP): adds $\lambda(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2$ to the discriminator loss, interpolating along the data/generated manifold.
- *Progressive growing* (ProGAN): train at low resolution first (4×4), then progressively add higher-resolution layers. The model learns coarse structure before fine detail.
- *Unrolled GANs:* unroll the discriminator update k steps before computing the generator gradient, reducing oscillatory training dynamics.

Nash equilibrium analysis. The GAN objective is a zero-sum two-player game. The Nash equilibrium satisfies:

$$G^* = \arg\min_G \max_D V(D, G) \tag{1}$$

$$D^* = \arg\max_D V(D, G^*) \tag{2}$$

At the global Nash equilibrium, $p_G = p_{\text{data}}$ and $D^*(\mathbf{x}) = 1/2$ everywhere. This is the unique fixed point of the gradient dynamics. However, practical training uses simultaneous gradient ascent-descent which does not generally converge to Nash equilibria — it can cycle.

Mode collapse mechanics. Mode collapse occurs when G generates a limited subset of modes of p_{data}. The mechanism:

1. G learns one mode; D learns to reject it.

2. G switches to a different mode to fool D.
3. D adapts; the cycle continues.

This *mode hopping* leaves individual modes well-covered momentarily but never simultaneously. Remedies:

- *Minibatch discrimination* (*Salimans et al., 2016*): D receives features comparing the current sample to other samples in the batch, penalising G for producing repetitive outputs.
- *Unrolled GANs* (*Metz et al., 2017*): G is trained against a k-step lookahead of D's updates, stabilising the dynamics.
- *Mode-seeking explicit objectives:* VEEGAN, DCGAN with diverse training tricks.

Gradient vanishing in the JS formulation. When D is well-trained, $(p_{\text{data}} \| p_G) = \log 2$ if the supports don't overlap (typical for real images vs. generated images). In this regime:

$$\nabla_\theta \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))] \approx 0$$

The generator gradient vanishes. This is why Goodfellow proposed the non-saturating alternative $\mathbb{E}_{\mathbf{z}}[-\log D(G(\mathbf{z}))]$ in the original paper, and why WGAN replaced JS with Wasserstein distance.

Training instabilities: causes and diagnostics.

- *Oscillation:* loss curves oscillate without converging; FID fluctuates. Caused by learning rate imbalance.
- *D saturates:* D outputs near-constant values; G gradient vanishes.
- *G memorises:* generated samples identical to training data; FID improves but precision/recall diverge.

The two-time-scale update rule (TTUR, Heusel et al., 2017) uses different learning rates $\alpha_D \gg \alpha_G$ to keep D well-trained relative to G, stabilising convergence.

19.4 Spectral Normalisation for Discriminators

Lipschitz constraint. A K-Lipschitz function satisfies $\|f(\mathbf{x}) - f(\mathbf{y})\| \leq K\|\mathbf{x} - \mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y}$. For WGAN, we require $K = 1$.

Spectral normalisation (Miyato et al. 2018) divides each weight matrix by its spectral norm (largest singular value) $\sigma_1(\mathbf{W})$:

$$\bar{\mathbf{W}} = \mathbf{W}/\sigma_1(\mathbf{W})$$

This ensures $\|\bar{\mathbf{W}}\|_2 = 1$, making each layer 1-Lipschitz. The composition of 1-Lipschitz layers is 1-Lipschitz.

Power iteration. $\sigma_1(\mathbf{W})$ is computed efficiently by power iteration (1–2 iterations per step):

$$\hat{\mathbf{v}} \leftarrow \mathbf{W}^\top \hat{\mathbf{u}}/\|\mathbf{W}^\top \hat{\mathbf{u}}\|, \qquad \hat{\mathbf{u}} \leftarrow \mathbf{W}\hat{\mathbf{v}}/\|\mathbf{W}\hat{\mathbf{v}}\|, \qquad \sigma_1 \approx \hat{\mathbf{u}}^\top \mathbf{W}\hat{\mathbf{v}}$$

PART 20

Denoising Diffusion Probabilistic Models

Diffusion models define a forward noising process and learn to reverse it. We derive the DDPM training objective, the reverse process, the probability flow ODE, and classifier-free guidance — every equation from scratch.

Power iteration for σ_1. Computing the full SVD of $\mathbf{W}$ is $O(\min(m,n)^2 \max(m,n))$ — too expensive per forward pass. Power iteration estimates $\sigma_1 = \|\mathbf{W}\|_2$ cheaply:

1. Initialise $\tilde{\mathbf{v}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
2. Repeat: $\tilde{\mathbf{u}} = \mathbf{W}\tilde{\mathbf{v}}/\|\mathbf{W}\tilde{\mathbf{v}}\|$; $\tilde{\mathbf{v}} = \mathbf{W}^\top\tilde{\mathbf{u}}/\|\mathbf{W}^\top\tilde{\mathbf{u}}\|$.
3. Estimate: $\hat{\sigma}_1 = \tilde{\mathbf{u}}^\top\mathbf{W}\tilde{\mathbf{v}}$.

One iteration per forward pass is sufficient in practice: $\tilde{\mathbf{v}}$ is cached and updated incrementally.

Lipschitz continuity of the discriminator. The Wasserstein GAN objective requires D to be 1-Lipschitz: $|D(\mathbf{x}) - D(\mathbf{y})| \leq \|\mathbf{x} - \mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y}$. For a composition of linear layers $\mathbf{h}^{(l)} = \mathbf{W}^{(l)}\mathbf{h}^{(l-1)}$:

$$\text{Lip}(f) \leq \prod_{l=1}^{L} \|\mathbf{W}^{(l)}\|_2$$

Spectral normalisation sets $\|\mathbf{W}^{(l)}\|_2 = 1$ for each layer, guaranteeing $\text{Lip}(D) \leq 1$ for linear layers. With Lipschitz activation functions (ReLU has Lip $= 1$), the whole network is 1-Lipschitz.

Comparison with gradient penalty (WGAN-GP).

Method	Constraint type	Overhead
Weight clipping	Global parameter clamp	None
Gradient penalty	Soft penalty on $\|\nabla_{\hat{x}} D\|$	Extra forward+backward
Spectral norm	Hard per-layer Lipschitz	One power iteration

Spectral normalisation is the most efficient and is used in StyleGAN, BigGAN, and most modern GAN architectures.

Self-attention GAN (SA-GAN). Zhang et al. (2019) show that spectral normalisation combined with self-attention layers in both G and D significantly improves fidelity on large-scale image generation. Spectral normalisation stabilises the self-attention layers by preventing the attention maps from exploding, enabling the discriminator to enforce global coherence constraints on generated images.

Singular value decomposition perspective. For a linear layer $\mathbf{W} \in \mathbb{R}^{m \times n}$, the singular value decomposition is $\mathbf{W} = \mathbf{U}\square\mathbf{V}^\top$ where $\square = \text{diag}(\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0)$. Spectral normalisation keeps $\sigma_1 = 1$ while leaving $\sigma_2, \ldots, \sigma_{\min(m,n)}$ uncontrolled. This means the effective rank of $\mathbf{W}$ is preserved; only the maximum singular value is constrained. In contrast, weight clipping $\|\mathbf{W}_{ij}\|_\infty \leq c$ penalises all singular values, often causing the weight matrix to become rank-deficient.

Orthogonal regularisation. A complementary approach (Brock et al., 2019, BigGAN) adds an orthogonality penalty to the generator:

$$\mathcal{L}_{\text{orth}} = \lambda \|\mathbf{W}^\top\mathbf{W} - \mathbf{I}\|_F^2$$

This encourages all singular values to be equal to 1, rather than only the largest. Combined with spectral normalisation on D, orthogonal regularisation produces the most stable GAN training known.

Gradient flow through the discriminator. For the generator update, the gradient of the GAN loss flows through D:

$$\nabla_G \mathcal{L} = \nabla_G D(G(\mathbf{z})) = \left.\frac{\partial D}{\partial \mathbf{x}}\right|_{\mathbf{x}=G(\mathbf{z})} \cdot \frac{\partial G}{\partial \theta_G}$$

If D is 1-Lipschitz (via spectral normalisation), $\|\partial D/\partial \mathbf{x}\| \leq 1$, so the gradient signal entering G is bounded. This prevents the gradient explosion that causes training instability in vanilla GANs: the discriminator cannot produce arbitrarily large gradient norms even when it confidently rejects fakes.

20.1 The Forward Process

Diffusion models (Ho et al., 2020) are based on a stochastic process that gradually transforms data into Gaussian noise, and learn to reverse this process.

Forward Process (Diffusion)

The forward process $q(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T)$ adds Gaussian noise in T steps:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t;\ \sqrt{1-\beta_t}\,\mathbf{x}_{t-1},\ \beta_t \mathbf{I}\right)$$

with noise schedule $0 < \beta_1 < \beta_2 < \cdots < \beta_T < 1$.

Closed form. Defining $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, one obtains directly:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t;\ \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0,\ (1-\bar{\alpha}_t)\mathbf{I}\right)$$

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

This is the *reparametrisation formula*: sampling $\mathbf{x}_t$ from $\mathbf{x}_0$ requires only a single white noise sample.

Interpretation of the process. At $t = 0$: $\mathbf{x}_0$ = original data. At $t = T$: with $\bar{\alpha}_T \approx 0$, $\mathbf{x}_T \approx \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The data has been completely destroyed into Gaussian noise.

Example — Evolution of the Noise Level

With $T = 1000$ and linear schedule $\beta_t = 10^{-4} + (0.02 - 10^{-4}) \cdot t/(T-1)$:

t	$\sqrt{\bar{\alpha}_t}$ (signal)	$\sqrt{1-\bar{\alpha}_t}$ (noise)
0	1.000	0.000
100	0.924	0.383
300	0.720	0.694
600	0.371	0.929
900	0.078	0.997
1000	0.012	≈ 1.000

Cosine schedule. The linear schedule has a signal that decays too rapidly in the early steps. Nichol & Dhariwal (2021) propose the cosine schedule:

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \quad f(t) = \cos^2\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)$$

which gives a more gradual transition and improves the quality of generated images.

Markov properties. The forward process is a Markov process: $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \ldots) = q(\mathbf{x}_t|\mathbf{x}_{t-1})$. This greatly simplifies the mathematics of the reverse process (Section 20.2). **Posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$.** The forward process is not only $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ (Markovian) but also has a tractable posterior when conditioned on $\mathbf{x}_0$:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$$

where:

$$\tilde{\mu}_t = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t, \qquad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

This is derived by applying Bayes' theorem to the forward Markov chain: $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \propto q(\mathbf{x}_t|\mathbf{x}_{t-1})q(\mathbf{x}_{t-1}|\mathbf{x}_0)$, both Gaussians, giving a Gaussian product.

Interpretation. The posterior mean $\tilde{\mu}_t$ is a weighted average of $\mathbf{x}_0$ and $\mathbf{x}_t$:

- At $t \approx 1$: weights are nearly equal, $\tilde{\mu}_t \approx (\mathbf{x}_0 + \mathbf{x}_t)/2$.
- At $t \approx T$: $\bar{\alpha}_{t-1} \approx \bar{\alpha}_t \approx 0$, $\tilde{\mu}_t \approx \sqrt{\alpha_t}\mathbf{x}_t$ (mostly noisy).

This tractable posterior is what makes DDPM's reverse process derivable: the reverse step $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is trained to match $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ with $\mathbf{x}_0$ estimated from ϵ_θ.

Noise schedule design.

- *Linear schedule* (Ho et al., 2020): $\beta_t = \beta_1 + (t-1)(\beta_T - \beta_1)/(T-1)$, $\beta_1 = 10^{-4}$, $\beta_T = 0.02$. Simple; works for $T = 1000$.
- *Cosine schedule* (Nichol & Dhariwal, 2021): $\bar{\alpha}_t = \cos^2\left(\frac{t/T+s}{1+s} \cdot \frac{\pi}{2}\right)$. Avoids near-zero SNR at early timesteps; improves FID.
- *Learned schedule*: optimise the schedule jointly with θ using the VLB; used in improved DDPM.

20.2 The Reverse Process and ELBO

The reverse process. Generation proceeds by traversing the Markov chain backwards: starting from $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, sample each step:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1};\, \boldsymbol{\mu}_\theta(\mathbf{x}_t, t),\, \Sigma_\theta(\mathbf{x}_t, t))$$

The joint of the reverse process is:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

ELBO for DDPM. Applying the standard decomposition to the negative log-likelihood:

$$-\log p_\theta(\mathbf{x}_0) \leq \underbrace{-\mathbb{E}_q[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)]}_{\text{reconstruction}} + \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \,\|\, p(\mathbf{x}_T))}_{\text{prior matching}} + \underbrace{\sum_{t=2}^{T} \mathbb{E}_q[D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))]}_{\text{denoising matching}}$$

The forward process posterior. The key technical insight is that the posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is an analytic Gaussian:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\big(\mathbf{x}_{t-1};\, \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0),\, \tilde{\beta}_t \mathbf{I}\big)$$

with:

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t, \qquad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Denoising matching. The KL terms in the sum are between Gaussians:

$$D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) = \frac{1}{2\tilde{\beta}_t}\|\tilde{\boldsymbol{\mu}}_t - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 + \text{const}$$

Each ELBO term therefore asks the network to predict the mean $\tilde{\boldsymbol{\mu}}_t$ of the forward posterior. Ho et al. (2020) show that reparametrising in terms of the noise $\boldsymbol{\epsilon}$ (Section 20.3) leads to a simpler objective and more stable training.

Variances Σ_θ: fixed choice

Ho et al. set $\Sigma_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$ with fixed σ_t^2: either $\sigma_t^2 = \beta_t$ (optimal for $\mathbf{x}_0 \sim \mathcal{N}$) or $\sigma_t^2 = \tilde{\beta}_t$ (optimal for deterministic $\mathbf{x}_0$). In practice the two choices give similar results. Nichol & Dhariwal (2021) learn Σ_θ with a scalar parameter interpolated in log-space.

The generative model defines the reverse process:

$$p_\theta(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t) = \mathcal{N}\big(\mathbf{x}_{t-1};\, \square_\theta(\mathbf{x}_t, t),\, \tilde{\beta}_t \mathbf{I}\big)$$

The log-likelihood lower bound (ELBO) decomposes into:

$$-\mathcal{L} = \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \,|\, \mathbf{x}_0)}\right] = \underbrace{-D_{\mathrm{KL}}(q(\mathbf{x}_T \,|\, \mathbf{x}_0) \,\|\, p(\mathbf{x}_T))}_{\approx 0} + \sum_{t=2}^{T} \underbrace{-D_{\mathrm{KL}}(q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t, \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t))}_{L_{t-1}} + \underbrace{\log p_\theta(\mathbf{x}_0 \,|\, \mathbf{x}_1}_{L_0}$$

Key: the posterior $q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t, \mathbf{x}_0)$ is tractable:

$$q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\big(\mathbf{x}_{t-1};\, \tilde{\square}_t(\mathbf{x}_t, \mathbf{x}_0),\, \tilde{\beta}_t \mathbf{I}\big)$$

$$\tilde{\square}_t = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t, \qquad \tilde{\beta}_t = \frac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}$$

ELBO decomposition for diffusion. The negative ELBO of DDPM can be written as a sum of denoising terms plus boundary terms:

$$-\mathcal{L}_{\mathrm{ELBO}} = \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \,\|\, p(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^{T} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}$$

- L_T: constant, independent of θ (forward process has no parameters); approximately 0 if $\bar{\alpha}_T \approx 0$.
- L_{t-1}: KL between two Gaussians, has closed form. This is what DDPM minimises.

- L_0: reconstruction term for the final denoising step; treated as a discrete decoder for images.

Parametrising the reverse mean. Since $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ has mean $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0)$, and we can express $\mathbf{x}_0 = (\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon)/\sqrt{\bar{\alpha}_t}$, the reverse process mean:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$$

The network ϵ_θ predicts the noise ϵ added at step t. Equivalently, one can predict $\mathbf{x}_0$ directly (x-prediction) or the velocity $v = \sqrt{\bar{\alpha}_t}\epsilon - \sqrt{1-\bar{\alpha}_t}\mathbf{x}_0$ (v-prediction); all are equivalent but differ in numerical stability.

Variance of the reverse process. Ho et al. fix $\Sigma_\theta(\mathbf{x}_t, t) = \sigma_t^2\mathbf{I}$ (not learned). Nichol & Dhariwal (2021) learn the variance by interpolating between β_t and $\tilde{\beta}_t$: $\Sigma_\theta = \exp(\nu \log \beta_t + (1-\nu)\log\tilde{\beta}_t)$ where $\nu \in [0,1]$ is the network output. Learning the variance improves FID and enables fewer sampling steps.

20.3 The DDPM Training Objective

Ho et al. (2020) simplify the diffusion ELBO into a *noise prediction* objective that is practically identical to training a Gaussian denoiser.

ELBO of the diffusion process. The evidence $\log p_\theta(\mathbf{x}_0)$ decomposes as:

$$\text{ELBO} = \mathbb{E}_q\Bigg[\underbrace{\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} - \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\|p(\mathbf{x}_T))}_{L_T} - \sum_{t=2}^{T}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\|p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}\Bigg]$$

Analytic forward posterior. Thanks to the Gaussian structure:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\big(\mathbf{x}_{t-1};\, \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0),\, \tilde{\beta}_t\mathbf{I}\big)$$

$$\tilde{\boldsymbol{\mu}}_t = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t, \qquad \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Every diffusion process has an equivalent *deterministic* ODE with the same marginals (Song et al., 2021):

$$\frac{d\mathbf{x}}{dt} = -\frac{1}{2}\dot{\beta}(t)\left(\mathbf{x} + \nabla_{\mathbf{x}}\log p_t(\mathbf{x})\right)$$

where $\nabla_{\mathbf{x}}\log p_t(\mathbf{x})$ is the *score function*. The connection to the noise predictor:

$$\nabla_{\mathbf{x}}\log p_t(\mathbf{x}_t) = -\frac{\square_\theta(\mathbf{x}_t, t)}{\sqrt{1-\bar{\alpha}_t}}$$

The ODE allows exact likelihood computation and deterministic sampling.

DDPM form. The DDPM forward process corresponds to a Variance Preserving (VP) SDE:

$$d\mathbf{x} = -\frac{\beta(t)}{2}\mathbf{x}\,dt + \sqrt{\beta(t)}\,d\mathbf{w}$$

The corresponding PF-ODE is:

$$d\mathbf{x} = -\frac{\beta(t)}{2}\left[\mathbf{x} + \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] dt$$

Substituting the approximated score $\nabla_{\mathbf{x}} \log p_t \approx -\epsilon_\theta(\mathbf{x}_t, t)/\sigma_t$:

$$\frac{d\mathbf{x}}{dt} = -\frac{\beta(t)}{2}\left[\mathbf{x} - \frac{\epsilon_\theta(\mathbf{x}, t)}{\sigma_t}\right]$$

Advantages of the PF-ODE.

- **Deterministic sampling**: $\mathbf{x}_T \mapsto \mathbf{x}_0$ is exact and invertible.
- **Fast sampling**: DDIM uses 10–50 ODE steps vs. 1000 DDPM steps.
- **Exact likelihood**: via the continuity equation and Hutchinson's estimator.
- **Latent interpolation**: encode $\mathbf{x}_0 \to \mathbf{x}_T$, interpolate, decode.

SDE vs. ODE.

- **SDE (DDPM)**: stochastic, $T = 1000$ steps, more robust to discretisation errors.
- **ODE (DDIM)**: deterministic, ≥ 20 steps, faster but slightly lower quality.
- **DPM-Solver**: high-order ODE solver for diffusion, 10–20 steps, quality $\approx$ DDPM-1000.

20.4 The Probability Flow ODE

Song et al. (2021) show that every diffusion SDE has an equivalent *ordinary* differential equation (ODE) whose marginal densities $p_t(\mathbf{x})$ are identical to those of the SDE at all times t. This ODE, called the *probability flow ODE* (PF-ODE), enables exact likelihood computation and deterministic, invertible generation.

From SDE to ODE. Consider the general Itô SDE of a forward diffusion process:

$$d\mathbf{x} = f(\mathbf{x}, t)\, dt + g(t)\, d\mathbf{w}$$

where $f : \mathbb{R}^d \times [0,T] \to \mathbb{R}^d$ is the drift, $g : [0,T] \to \mathbb{R}$ is the diffusion coefficient, and $\mathbf{w}$ is a standard Wiener process. The Fokker–Planck equation governs the evolution of the marginal density p_t:

$$\frac{\partial p_t}{\partial t} = -\nabla \cdot (p_t f) + \frac{g(t)^2}{2} \Delta p_t$$

Using the identity $\Delta p_t = \nabla \cdot (p_t \, \nabla \log p_t)$, this can be rewritten as:

$$\frac{\partial p_t}{\partial t} = -\nabla \cdot \left(p_t \left[f(\mathbf{x}, t) - \frac{g(t)^2}{2} \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right] \right)$$

This is the continuity equation $\partial_t p_t + \nabla \cdot (p_t \, \tilde{v}) = 0$ for the deterministic velocity field:

$$\tilde{v}(\mathbf{x}, t) = f(\mathbf{x}, t) - \frac{g(t)^2}{2} \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

Probability Flow ODE

The PF-ODE whose marginals match the SDE marginals p_t for all $t \in [0,T]$:

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, t) - \frac{g(t)^2}{2} \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

This is a *deterministic* ODE. Solving it forward from $\mathbf{x}_0 \sim p_0$ produces $\mathbf{x}_T \sim p_T = \mathcal{N}(\mathbf{0}, \mathbf{I})$ (encoding). Solving it backward from $\mathbf{x}_T \sim p_T$ produces $\mathbf{x}_0 \sim p_0$ (generation).

Instantiation for DDPM. In the DDPM variance-preserving (VP) SDE, $f(\mathbf{x}, t) = -\frac{\beta(t)}{2}\mathbf{x}$ and $g(t) = \sqrt{\beta(t)}$. The PF-ODE becomes:

$$\frac{d\mathbf{x}}{dt} = -\frac{\beta(t)}{2} [\mathbf{x} + \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]$$

Substituting the learned score approximation $\nabla_{\mathbf{x}} \log p_t \approx -\epsilon_\theta(\mathbf{x}_t, t)/\sigma_t$:

$$\frac{d\mathbf{x}}{dt} = -\frac{\beta(t)}{2} \left[\mathbf{x} - \frac{\epsilon_\theta(\mathbf{x}, t)}{\sigma_t} \right]$$

This is the "neural" PF-ODE solved by DDIM and DPM-Solver.

Exact likelihood via the instantaneous change-of-variables formula. The log-likelihood under the PF-ODE flow decomposes as:

$$\log p_0(\mathbf{x}_0) = \log p_T(\mathbf{x}_T) + \int_0^T \nabla \cdot \tilde{v}(\mathbf{x}(t), t)\, dt$$

where $\nabla \cdot \tilde{v} = \mathrm{tr}(\partial \tilde{v}/\partial \mathbf{x})$ is the divergence. For a d-dimensional neural network, the exact trace costs $O(d^2)$. The **Hutchinson estimator** reduces this to $O(d)$:

$$\mathrm{tr}\left(\frac{\partial \tilde{v}}{\partial \mathbf{x}}\right) = \mathbb{E}_{\square}\left[\square^\top \frac{\partial \tilde{v}}{\partial \mathbf{x}} \square\right], \quad \square \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Each evaluation of $\square^\top (\partial \tilde{v}/\partial \mathbf{x})$ is a single vector-Jacobian product (one backward pass). Combined with a numerical ODE solver with N_{steps} function evaluations, total cost is $O(N_{\text{steps}} \cdot d)$.

Example — Likelihood computation on CIFAR-10

For a DDPM trained on CIFAR-10 ($d = 3072$, $T = 1000$):

- **SDE sampling:** 1000 Euler–Maruyama steps; stochastic; cannot compute likelihood.
- **PF-ODE sampling:** 50–200 steps of RK45; deterministic; log-likelihood = $\log p_T(\mathbf{x}_T) + \int_0^T \mathrm{tr}(\partial \tilde{v}/\partial \mathbf{x})\, dt$.
- **Reported NLL:** ≈ 3.0 bits/dim (Song et al., 2021), competitive with autoregressive models.

Invertibility and latent manipulation. Because the PF-ODE is a deterministic invertible map $\Phi : [0 \to T] : \mathbf{x}_0 \mapsto \mathbf{x}_T$, it enables:

- **Image editing:** encode $\mathbf{x}_0 \to \mathbf{x}_T$ (inversion), modify $\mathbf{x}_T$ in latent space, decode $\mathbf{x}_T \to \mathbf{x}_0'$.
- **Interpolation:** encode two images to $\mathbf{x}_T^{(1)}$ and $\mathbf{x}_T^{(2)}$, interpolate spherically $\mathbf{x}_T(\lambda) = (1-\lambda)\mathbf{x}_T^{(1)} + \lambda \mathbf{x}_T^{(2)}$, decode.
- **Semantic arithmetic:** $\mathbf{x}_T^{\text{cat}} - \mathbf{x}_T^{\text{dog}} + \mathbf{x}_T^{\text{wolf}} \approx \mathbf{x}_T^{\text{wolf-cat}}$.

These operations are not possible with stochastic SDE sampling, since different runs of the reverse SDE from the same $\mathbf{x}_T$ give different $\mathbf{x}_0$.

Variance-exploding (VE) SDE. The NCSN model corresponds to $f = 0$, $g(t) = \sqrt{d[\sigma^2(t)]/dt}$ (variance grows from σ_0 to σ_T). Its PF-ODE:

$$\frac{d\mathbf{x}}{dt} = -\frac{1}{2}\frac{d\sigma^2}{dt} \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

At $t = 0$: $\mathbf{x}$ is data; at $t = T$: $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \sigma_T^2 \mathbf{I})$. With score approximation, the ODE is solved backward from $t = T$ to $t = 0$.

Sub-VP SDE. The sub-VP SDE (Song et al., 2021) uses $f(\mathbf{x}, t) = -\frac{1}{2}\beta(t)\mathbf{x}$ and $g(t) = \sqrt{\beta(t)(1 - e^{-2\int_0^t \beta(s)ds})}$, giving a marginal distribution with the same mean as VP but smaller variance. Sub-VP achieves better likelihood estimates while maintaining sample quality comparable to VP.

DPM-Solver. Lu et al. (2022) observe that the PF-ODE for diffusion models has a semi-linear structure that can be exploited for higher-order solvers:

$$\mathbf{x}_s = e^{-(h_s - h_t)/2}\mathbf{x}_t + (\text{integral of noise prediction over } [s, t])$$

where $h_t = \int_0^t \beta(\tau)d\tau/2$. DPM-Solver-3 uses a 3rd-order Taylor expansion of the integral, achieving FID ≈ 3.2 on CIFAR-10 in only 12 function evaluations — a 100× speedup over DDPM-1000.

20.5 Classifier-Free Guidance — Complete Derivation

Classifier-free guidance conditions the generation on a label $\mathbf{c}$ without a separate classifier.

Train a single network $\square_\theta(\mathbf{x}_t, t, \mathbf{c})$ that is conditioned on $\mathbf{c}$ some of the time and unconditional (with $\mathbf{c} = \emptyset$) the rest.

Score guidance. At inference, use the guided score:

$$\hat{\square}_\theta(\mathbf{x}_t, t, \mathbf{c}) = (1 + w)\, \square_\theta(\mathbf{x}_t, t, \mathbf{c}) - w\, \square_\theta(\mathbf{x}_t, t, \emptyset)$$

where $w \geq 0$ is the *guidance scale*.

Derivation. The optimal conditional score is:

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x} \,|\, \mathbf{c}) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(\mathbf{c} \,|\, \mathbf{x})$$

Amplifying the classifier term by w:

$$\nabla_{\mathbf{x}} \log p_t^w(\mathbf{x} \,|\, \mathbf{c}) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + (1 + w)\, \nabla_{\mathbf{x}} \log p_t(\mathbf{c} \,|\, \mathbf{x})$$

Substituting the score-to-noise conversion:

$$\hat{\square} = \square_\theta(\mathbf{x}_t, t, \emptyset) + (1 + w)\, [\square_\theta(\mathbf{x}_t, t, \mathbf{c}) - \square_\theta(\mathbf{x}_t, t, \emptyset)] = (1 + w)\square_\theta(\cdot, \mathbf{c}) - w\square_\theta(\cdot, \emptyset)$$

Example — Guidance effect at $w = 3$

Suppose $\square_\theta(\cdot, \mathbf{c}) = 0.4$ and $\square_\theta(\cdot, \emptyset) = 0.1$ (scalar example).
$\hat{\square} = (1 + 3) \cdot 0.4 - 3 \cdot 0.1 = 1.6 - 0.3 = 1.3$
The guided noise prediction is $3.25\times$ the conditional prediction — the reverse step is pulled strongly toward samples consistent with label $\mathbf{c}$. This trades diversity for sample quality.

Why guidance improves quality at the cost of diversity. Recall the conditional reverse SDE has drift proportional to $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|y)$. By Bayes:

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|y) = \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p_t(y|\mathbf{x}_t)$$

CFG with scale w approximates w times the conditional score gradient:

$$\tilde{\epsilon}_\theta(\mathbf{x}_t, t, y) = \epsilon_\theta(\mathbf{x}_t, t, \emptyset) + w(\epsilon_\theta(\mathbf{x}_t, t, y) - \epsilon_\theta(\mathbf{x}_t, t, \emptyset))$$

$w = 1$: standard conditional model. $w > 1$: extrapolate beyond $p(y|\mathbf{x}_t)$, sharpening the conditioning at the cost of diversity (Precision-Recall tradeoff). $w = 7$–9 is typical for text-to-image.

FID vs. CLIP score tradeoff. Increasing w improves CLIP score (faithfulness to text prompt) but increases FID (reduces sample diversity). This is a fundamental tradeoff: at $w \to \infty$ the model produces the single most-likely image given the prompt — high fidelity, zero diversity.

Classifier guidance (CG) vs. classifier-free guidance (CFG). CG (Dhariwal & Nichol, 2021) uses a separately trained noisy classifier $p_\phi(\mathbf{c}|\mathbf{x}_t)$ to steer generation:

$$\hat{s}(\mathbf{x}_t, t, \mathbf{c}) = s_\theta(\mathbf{x}_t, t) + w\, \nabla_{\mathbf{x}_t} \log p_\phi(\mathbf{c}|\mathbf{x}_t)$$

CFG avoids training a separate classifier. CFG advantages: (1) only one model to train; (2) guidance is aligned with the generative model's score, avoiding distribution shift; (3) scales

naturally to text conditioning. CG advantage: guidance can be applied to a pre-trained unconditional model without retraining.

Dynamic thresholding. At high w, samples drift outside the training distribution, producing over-saturated images. Imagen (Saharia et al., 2022) introduces *dynamic thresholding*: at each reverse step, clip the predicted $\hat{\mathbf{x}}_0 = (\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon})/\sqrt{\bar{\alpha}_t}$ to the p-th percentile of absolute values:

$$\hat{\mathbf{x}}_0 \leftarrow \text{clip}(\hat{\mathbf{x}}_0,\ -s,\ s)\,/s, \quad s = \max(\text{percentile}_p(|\hat{\mathbf{x}}_0|),\ 1)$$

with $p = 99.5$. This confines predictions to $[-1, 1]$ and prevents pixel saturation, enabling $w = 7.5$ without artefacts on text-to-image tasks.

Guidance in flow matching. CFG applies identically to flow matching (ODE case):

$$\hat{v}_\theta(\mathbf{x}_t, t, \mathbf{c}) = v_\theta(\mathbf{x}_t, t, \emptyset) + w[v_\theta(\mathbf{x}_t, t, \mathbf{c}) - v_\theta(\mathbf{x}_t, t, \emptyset)]$$

The velocity is linearly extrapolated in the direction of the class-conditional flow. Stable Diffusion 3 and Flux use this formulation with $w \in [3, 7]$.

Example — Guidance in Stable Diffusion

1.5 default: $w = 7.5$. Prompt: "a photo of a cat".
At $w = 1$: diverse cats (various poses, breeds, backgrounds), some may not resemble real photos.
At $w = 7.5$: photorealistic cats, more consistent with the prompt style, less variety in pose.
At $w = 20$: very sharp, saturated images; artefacts; all look nearly identical.
In practice, $w \in [5, 12]$ gives the best balance.

20.6 Score-Based Generative Models

Song & Ermon (2019) derive diffusion models from a different angle: learning the *score function* $\nabla_{\mathbf{x}} \log p(\mathbf{x})$.

Langevin dynamics. Samples from $p(\mathbf{x})$ can be generated by:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \frac{\epsilon}{2}\nabla_{\mathbf{x}} \log p(\mathbf{x}_t) + \sqrt{\epsilon}\,\mathbf{z}_t, \qquad \mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

As $\epsilon \to 0$, the stationary distribution is $p(\mathbf{x})$.

Score matching. Train a network $s_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$ by minimising:

$$\mathbb{E}_p\left[\tfrac{1}{2}\|s_\theta(\mathbf{x})\|^2 + \text{tr}(\nabla_{\mathbf{x}} s_\theta(\mathbf{x}))\right]$$

The trace term $\text{tr}(\nabla s_\theta)$ is expensive for high d. **Sliced score matching** and **denoising score matching** are tractable alternatives.

Connection to DDPM. The DDPM denoising network ϵ_θ predicts the noise added at step t. This is equivalent to learning the conditional score:

$$s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) = -\frac{\square}{\sqrt{1-\bar{\alpha}_t}}$$

where $\square$ is the added noise. DDPM and score-based models are mathematically equivalent.

Noise-conditional score network (NCSN). To handle the low-density problem (score is undefined far from the data manifold), Song & Ermon (2019) train a single network $s_\theta(\mathbf{x}, \sigma)$ conditioned on noise level σ:

$$\ell(\theta) = \sum_i \lambda(\sigma_i)\, \mathbb{E}_{\tilde{\mathbf{x}}}\left[\tfrac{1}{2}\left\| s_\theta(\tilde{\mathbf{x}}, \sigma_i) - \nabla_{\tilde{\mathbf{x}}} \log p_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x})\right\|^2\right]$$

where $p_{\sigma_i}(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}; \mathbf{x}, \sigma_i^2 \mathbf{I})$ and $\nabla_{\tilde{\mathbf{x}}} \log p_{\sigma_i} = -(\tilde{\mathbf{x}} - \mathbf{x})/\sigma_i^2$ is known analytically.

Unified view. DDPM (§20.3) and score-based models are equivalent: the DDPM noise prediction network $\epsilon_\theta(\mathbf{x}_t, t)$ is related to the score by:

$$s_\theta(\mathbf{x}_t, t) = -\frac{\epsilon_\theta(\mathbf{x}_t, t)}{\sigma_t}$$

The DDPM objective $\|\epsilon_\theta - \epsilon\|^2$ is equivalent to denoising score matching. This unification (Song et al., 2021, "Score-Based Generative Modeling through Stochastic Differential Equations") frames all diffusion models as learning to reverse an SDE: $d\mathbf{x} = f(\mathbf{x}, t)dt + g(t)d\mathbf{w}$ with reverse SDE $d\mathbf{x} = [f - g^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]dt + g\, d\bar{\mathbf{w}}$.

Annealed Importance Sampling (AIS) for likelihood estimation. For trained score models, exact likelihood computation requires the normalisation constant Z_θ. AIS provides an unbiased estimator by constructing a chain of distributions from $p_{\sigma_{\max}}(\mathbf{x}) \approx \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$ to $p_{\sigma_{\min}}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$:

$$\log Z = \log Z_0 + \sum_{t=1}^{T} \log \frac{Z_t}{Z_{t-1}} \approx \log Z_0 + \sum_{t=1}^{T} \mathbb{E}\left[\log \frac{p_t(\mathbf{x}^{(t)})}{p_{t-1}(\mathbf{x}^{(t)})}\right]$$

Numerically, NCSN achieves competitive likelihood estimates on CIFAR-10 (≈ 3.2 bpd) using AIS with 1000 annealing steps.

Consistency of DSM. Denoising Score Matching is a consistent estimator: as $S \to \infty$ and $\sigma \to 0$, the DSM minimiser converges to the true score $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. The convergence rate depends on the smoothness of p and the neural network approximation class. For finite σ, DSM estimates the score of the smoothed distribution $p_\sigma = p * \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$, which explains why multi-scale training (with σ ranging from large to small) is necessary for accuracy at all scales.

Langevin MCMC for sampling. Given the learned score $s_\theta(\mathbf{x}, \sigma)$, samples are generated via annealed Langevin dynamics: starting from $\mathbf{x}^{(0)} \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$, for each noise level σ_i run T steps:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \frac{\epsilon_i}{2} s_\theta(\mathbf{x}^{(t)}, \sigma_i) + \sqrt{\epsilon_i}\, \mathbf{z}^{(t)}, \quad \mathbf{z}^{(t)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

The step size $\epsilon_i = \epsilon_0 (\sigma_i / \sigma_1)^2$ scales with σ_i^2 to maintain a fixed signal-to-noise ratio. As $\epsilon_i \to 0$ and $T \to \infty$, the chain converges to $p_{\sigma_i}(\mathbf{x})$.

Predictor-Corrector samplers. Song et al. (2021) combine a numerical ODE/SDE solver ("predictor") with Langevin MCMC steps ("corrector"). At each time step:

1. **Predictor:** advance $\mathbf{x}_t \to \mathbf{x}_{t-\Delta t}$ using a reverse-SDE solver or the PF-ODE.
2. **Corrector:** run M steps of Langevin dynamics at fixed t to sharpen the distribution.

This improves sample quality by correcting discretisation errors in the predictor. With $N = 1000$ predictor steps and $M = 1$ corrector step per level, PC samplers achieve FID ≈ 2.2 on CIFAR-10 — better than DDPM at comparable NFE.

Score network architecture. The score network $s_\theta(\mathbf{x}, t) : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ must output a vector of the same dimensionality as $\mathbf{x}$. For images:

- **NCSNv2** (Song & Ermon, 2020): U-Net with instance normalisation; spectral normalisation on all layers; time t encoded as a scalar input.
- **DDPM U-Net** (Ho et al., 2020): U-Net with group normalisation; sinusoidal time embedding $[\cos(t/\omega_k), \sin(t/\omega_k)]_{k=1}^{d}$ injected at each residual block via FiLM conditioning.
- **DiT** (Peebles & Xie, 2023): Vision Transformer with adaptive layer normalisation for time conditioning; scales to ImageNet 512×512.

Score parameterisation and ϵ-prediction equivalence. Three equivalent parameterisations:

Parameterisation	Network output	Relation to score
ϵ-prediction	$\epsilon_\theta(\mathbf{x}_t, t) \approx \square$	$s_\theta = -\epsilon_\theta / \sigma_t$
$\mathbf{x}_0$-prediction	$\hat{\mathbf{x}}_0(\mathbf{x}_t, t) \approx \mathbf{x}_0$	$s_\theta = (\sqrt{\bar\alpha_t}\hat{\mathbf{x}}_0 - \mathbf{x}_t)/(1-\bar\alpha_t)$
v-prediction	$v_\theta = \sqrt{\bar\alpha_t}\square - \sqrt{1-\bar\alpha_t}\mathbf{x}_0$	used by Imagen Video

All three are equivalent at optimum; they differ in the weighting of loss terms across noise levels, affecting training dynamics and sample quality for different noise schedules.

20.7 DDIM: Deterministic Sampling

Denoising Diffusion Implicit Models (Song et al. 2020) derive a family of non-Markovian inference processes sharing the same marginals $q(\mathbf{x}_t|\mathbf{x}_0)$ as DDPM but with deterministic (or partially stochastic) reverse steps:

$$\mathbf{x}_{t-1} = \sqrt{\bar\alpha_{t-1}} \underbrace{\left(\frac{\mathbf{x}_t - \sqrt{1-\bar\alpha_t}\,\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\bar\alpha_t}} \right)}_{\text{“predicted } \mathbf{x}_0\text{”}} + \sqrt{1 - \bar\alpha_{t-1} - \sigma_t^2}\,\epsilon_\theta(\mathbf{x}_t, t) + \sigma_t \mathbf{z}_t$$

Setting $\sigma_t = 0$ for all t: fully deterministic — same noise $\mathbf{z}$ always generates the same image. This enables:

- Consistent generation (same seed $\to$ same image).
- Deterministic latent encoding (real images can be “inverted” to noise).
- Sampling in 10–50 steps instead of 1000.

Classifier-Free Guidance (CFG): meccanismo. Ho & Salimans (2022): invece di addestrare un classificatore separato, si addestra un singolo modello condizionale $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{c})$ che gestisce sia la generazione condizionata (con label $\mathbf{c}$) che quella non condizionata (con $\mathbf{c} = \emptyset$, label nullo):

$$\boldsymbol{\epsilon}_{\text{guided}} = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \emptyset) + w \cdot [\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{c}) - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \emptyset)]$$

where $w > 0$ is the *guidance scale*. For $w = 0$: unconditional generatata. Per $w = 1$: predizione condizionata standard. Per $w > 1$: amplificazione del condizionamento (alta fedeltà al prompt, minore diversità).

Trade-off fedeltà-diversità. Il guidance scale w controlla il trade-off:

- w grande ($\approx$ 7–20): immagini molto fedeli al prompt, poca varietà, possibile saturazione.
- w piccolo ($\approx$ 1–3): maggiore diversità, meno allineamento al prompt.
- Optimal w dipende dal task: $w \approx 7.5$ per testo-immagine (Stable Diffusion default).

Implementazione pratica. Si usano due forward pass per step: uno condizionato, uno non condizionato. In Stable Diffusion questo è il collo di bottiglia di velocità; tecniche come il batching dei due forward pass e la pairwise attention optimization (PAB) riducono il costo a $\approx 1.5\times$ rispetto a un singolo forward pass.

PART 21

Flow Matching

Flow matching learns a vector field that transports a source distribution to a target. We derive conditional flow matching (CFM) and the optimal transport variant (OT-CFM).

Why fewer steps? DDPM requires $T = 1000$ steps because each step is a small Gaussian perturbation. DDIM reinterprets the same trained ϵ_θ as computing a trajectory in "predicted $\mathbf{x}_0$" space:

$$\hat{\mathbf{x}}_0^{(t)} = \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}}$$

The DDIM update moves along the line from $\mathbf{x}_t$ toward $\hat{\mathbf{x}}_0^{(t)}$, allowing large steps of size $\Delta t \sim 50$–250 without retraining. With $\sigma_t = 0$ (fully deterministic), this defines an ODE trajectory.

Inversion. Since DDIM is deterministic, it is invertible: given a sample $\mathbf{x}_0$, we can compute the corresponding noise $\mathbf{x}_T = f^{-1}(\mathbf{x}_0)$ by running the reverse ODE backwards. This enables:

- *Image editing:* invert a real image to $\mathbf{x}_T$, change the conditioning (text prompt), then decode. Used in Prompt-to-Prompt, Null-Text Inversion.
- *Interpolation:* linearly interpolate between $\mathbf{x}_T^{(1)}$ and $\mathbf{x}_T^{(2)}$ in latent space, then decode — smooth interpolation in image space.

EXAMPLE — DDIM vs. DDPM Sampling

Method	Steps	FID (CIFAR-10)	Deterministic	Inversion
DDPM	1000	3.17	No	No
DDIM	100	4.16	Yes	Yes
DDIM	50	4.67	Yes	Yes
DDIM	20	6.84	Yes	Yes

$50\times$ speedup with modest quality loss. For latent diffusion (Stable Diffusion), $\sim$20 DDIM steps suffices for high-quality 512×512 image generation in $\sim$3 seconds on an A100.

21.1 Continuous Normalising Flows and the Continuity Equation

Idea. Invece di comporre un numero finito di trasformazioni invertibili, i Continuous Normalizing Flows (CNF, Chen et al. 2018) definiscono una traiettoria continua nel tempo tramite un'ODE:

$$\frac{d\mathbf{z}(t)}{dt} = \mathbf{v}_\theta(\mathbf{z}(t), t), \qquad t \in [0,1]$$

where $\mathbf{v}_\theta : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ is a velocity field parametrizzato da una rete neurale.

Equazione di continuità. L'evoluzione della densità $p_t(\mathbf{z})$ lungo il campo $\mathbf{v}$ è governata dall'equazione di continuità (conservazione della probabilità):

$$\frac{\partial p_t}{\partial t} + \nabla \cdot (p_t\, \mathbf{v}_\theta) = 0$$

Integrating this equation, the log-density changes as:

$$\frac{d \log p_t(\mathbf{z}(t))}{dt} = -\nabla \cdot \mathbf{v}_\theta(\mathbf{z}(t), t) = -\mathrm{tr}\left(\frac{\partial \mathbf{v}_\theta}{\partial \mathbf{z}}\right)$$

Calcolo della log-likelihood. Partendo da $\mathbf{z}(1) \sim p_1$ (gaussiana base) e integrando all'indietro:

$$\log p_0(\mathbf{z}(0)) = \log p_1(\mathbf{z}(1)) + \int_0^1 \mathrm{tr}\left(\frac{\partial \mathbf{v}_\theta}{\partial \mathbf{z}}(\mathbf{z}(t), t)\right) dt$$

Computing the trace of the Jacobian $\partial \mathbf{v}/\partial \mathbf{z}$ richiede $O(d^2)$ con backpropagation. L'approssimazione stocastica di Hutchinson usa $\mathrm{tr}(A) \approx \boldsymbol{\epsilon}^\top A \boldsymbol{\epsilon}$ con $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, riducendo a $O(d)$.

Solver ODE per CNF

ODE integration uses adaptive solvers (e.g. Dormand-Prince RK45). The number of function evaluations of $\mathbf{v}_\theta$ (NFE) is proportional to the "stiffness" of the trajectory. For samplo veloce: tochastic interpolants o ODE rette (conditional flow matching, Sezione 21.2) riducono NFE a ≤ 10.

Backpropagation attraverso la ODE: metodo adjoint. Calcolare $\nabla_\theta \mathcal{L}$ richiede backpropagare attraverso il solver ODE. Il metodo adjoint (Pontryagin) evita di memorizzare l'intera traiettoria: introduce un'ODE aggiuntiva per lo stato aggiunto $\mathbf{a}(t) = \partial \mathcal{L}/\partial \mathbf{z}(t)$ e risolve le due ODE (forward e adjoint) sequenzialmente, con costo di memoria $O(1)$ rispetto alla profondità di integrazione.

A continuous normalising flow (CNF) defines a time-dependent vector field $v_\theta(\mathbf{x}, t) : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ generating an ODE:

$$\frac{d\mathbf{x}}{dt} = v_\theta(\mathbf{x}(t), t), \qquad \mathbf{x}(0) \sim p_0$$

The density p_t evolves according to the *continuity equation*:

$$\frac{\partial p_t}{\partial t} + \mathrm{div}(p_t v_\theta) = 0$$

Goal: learn v_θ such that $p_1 \approx p_{\text{data}}$.

Divergence computation. The instantaneous change-of-variables formula requires $\mathrm{tr}(\partial \mathbf{v}/\partial \mathbf{z})$. For a general $\mathbf{v}_\theta : \mathbb{R}^d \to \mathbb{R}^d$, exact computation costs $O(d^2)$ (full Jacobian). The Hutchinson estimator reduces this to $O(d)$:

$$\mathrm{tr}\left(\frac{\partial \mathbf{v}}{\partial \mathbf{z}}\right) = \mathbb{E}_{\square}\left[\square^\top \frac{\partial \mathbf{v}}{\partial \mathbf{z}} \square\right], \quad \square \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \text{ or Rademacher}$$

Only one vector-Jacobian product $\square^\top \frac{\partial \mathbf{v}}{\partial \mathbf{z}}$ is needed, costing one backward pass: $O(d)$.

Neural ODE. A CNF is built on a Neural ODE: a continuous-depth model where the hidden state evolves according to $d\mathbf{z}/dt = f_\theta(\mathbf{z}(t), t)$. The forward pass integrates this ODE from $t = 0$ to $t = 1$ using an adaptive solver (Dormand-Prince RK45). Backpropagation uses the adjoint method to avoid storing intermediate activations:

$$\frac{d\mathbf{a}}{dt} = -\mathbf{a}(t)^\top \frac{\partial f_\theta}{\partial \mathbf{z}}, \quad \mathbf{a}(t) = \frac{\partial \mathcal{L}}{\partial \mathbf{z}(t)}$$

Memory cost: $O(d)$ (just the adjoint), vs. $O(T \cdot d)$ for storing all ODE steps.

Limitation. CNFs are expressive but slow: ODE integration at test time requires hundreds of function evaluations. Flow Matching (§21.2) addresses this by replacing the ODE with a direct regression target.

Neural ODE as CNF. A *continuous normalising flow* defines a trajectory $\mathbf{z}(t)$ governed by a neural ODE:

$$\frac{d\mathbf{z}}{dt} = v_\theta(\mathbf{z}(t), t), \qquad \mathbf{z}(0) \sim p_0(\mathbf{z})$$

where $v_\theta : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ is a learned velocity field. The probability density evolves according to the *continuity equation*:

$$\frac{\partial p_t}{\partial t} = -\div (p_t v_\theta)$$

which is equivalent to the instantaneous change-of-variables formula:

$$\log p_t(\mathbf{z}(t)) = \log p_0(\mathbf{z}(0)) - \int_0^t \div v_\theta(\mathbf{z}(s), s)\, ds$$

The log-density change is the integral of the divergence of the velocity field along the trajectory.

FFJORD and Hutchinson's trace estimator. Computing $\div v_\theta = (\nabla v_\theta)$ exactly requires d Jacobian-vector products — $O(d^2)$ for general v_θ. Grathwohl et al. (2018) use Hutchinson's estimator:

$$(\mathbf{A}) = \mathbb{E}_{\square \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[\square^\top \mathbf{A} \square]$$

Applied to $\mathbf{A} = \nabla v_\theta$: $\div v_\theta \approx \square^\top (\nabla_{\mathbf{z}} v_\theta) \square$ requires only one JVP ($O(d)$ with auto-diff). This is the FFJORD trick: free-form Jacobian-based density estimation via Ordinary differential equations.

Training CNFs. Two strategies:

1. *Maximum likelihood:* maximise $\sum_n \log p_1(\mathbf{x}^{(n)})$ directly; requires integrating the ODE forward to get $\mathbf{z}(1) = \mathbf{x}$ from $\mathbf{z}(0) \sim p_0$ and tracking the log-density integral. Expensive (multiple ODE solves per batch).
2. *Flow matching:* avoid simulating the ODE during training by matching the velocity field to target flows (§21.2). Training is simulation-free.

21.2 Conditional Flow Matching Objective

The CNF training problem. The MLE objective for CNFs requires simulating the forward ODE during training to compute the likelihood, making training expensive (many evaluations of $\mathbf{v}_\theta$ per example).

Flow Matching (Lipman et al., 2022). Flow Matching trains a velocity field $\mathbf{v}_\theta(\mathbf{x}, t)$ to generate a flow transforming p_0 (noise) into p_1 (data), without simulating the ODE during training. The objective:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t \sim \mathcal{U}[0,1],\, \mathbf{x} \sim p_t}\left[\|\mathbf{v}_\theta(\mathbf{x}, t) - \mathbf{u}_t(\mathbf{x})\|^2\right]$$

where $\mathbf{u}_t(\mathbf{x})$ is the marginal velocity field generating p_t.

Relationship to diffusion models. CFM generalises diffusion models: choosing $p_0 = \mathcal{N}(\mathbf{0}, \mathbf{I})$ with a VP or VE noise schedule recovers the weighted score-matching objective. The key

difference: in CFM trajectories are straight (straight ODEs), so sampling requires few steps (NFE ≈ 10) vs. the 1000 steps of DDPM.

Advantages of CFM.

- Training without ODE simulation: each gradient step is $O(d)$ like a standard network.
- Straight trajectories $\Rightarrow$ simple ODE solvers and few steps.
- Easily extended to discrete data, data on manifolds (Riemannian FM), and sequences.

Flow Matching (FM) objective (Lipman et al., 2022): regress v_θ onto a *target vector field* u_t:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,\mathbf{x}\sim p_t}\left[\|v_\theta(\mathbf{x},t) - u_t(\mathbf{x})\|^2\right]$$

The target u_t is the unique vector field that generates the probability path p_t. However, u_t requires knowledge of the full marginal p_t, which is intractable.

Key theorem. The gradients of $\mathcal{L}_{\text{FM}}$ and the *Conditional* FM objective $\mathcal{L}_{\text{CFM}}$ are equal:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,\mathbf{z},\mathbf{x}\sim p_t(\mathbf{x}\,|\,\mathbf{z})}\left[\|v_\theta(\mathbf{x},t) - u_t(\mathbf{x}\,|\,\mathbf{z})\|^2\right]$$

Here $\mathbf{z}$ is a conditioning variable (e.g. the data point), and $u_t(\mathbf{x}\,|\,\mathbf{z})$ is the conditional vector field — tractable by construction.

Why FM outperforms diffusion score matching. Score matching requires learning $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, which involves the marginal density p_t — hard to estimate for intermediate t. CFM instead directly specifies *conditional* vector fields $u_t(\mathbf{x}|\mathbf{x}_1)$ that generate simple conditional flows (straight lines from $\mathbf{x}_0$ to $\mathbf{x}_1$). The marginal vector field:

$$u_t(\mathbf{x}) = \mathbb{E}_{p(\mathbf{x}_1|\mathbf{x})}[u_t(\mathbf{x}|\mathbf{x}_1)]$$

generates the marginal flow p_t (proof: continuity equation). Since we only train on conditional targets, the regression problem is much simpler (lower variance).

Geometric intuition. For the Gaussian conditional path $p_t(\mathbf{x}|\mathbf{x}_1) = \mathcal{N}(\mathbf{x}; t\mathbf{x}_1, (1-(1-\sigma_{\min})t)^2\mathbf{I})$, the conditional flow field is:

$$u_t(\mathbf{x}|\mathbf{x}_1) = \frac{\mathbf{x}_1 - (1-\sigma_{\min})\mathbf{x}}{1-(1-\sigma_{\min})t}$$

At $t=0$: points along the direction toward $\mathbf{x}_1$. At $t=1$: $u_1 = 0$ (arrived). This is a straight-line interpolation in distribution space, which ODE solvers can traverse in very few steps.

Connections.

- With $\sigma_{\min} \to 0$, CFM recovers the score matching objective of DDPM (up to a time reparameterisation).
- Stable Diffusion 3 and Flux use FM as their training objective (replacing DDPM), achieving better quality and faster sampling.

Derivation of the CFM equivalence. The key theorem (Lipman et al., 2022, Theorem 1) states $\nabla_\theta \mathcal{L}_{\text{FM}} = \nabla_\theta \mathcal{L}_{\text{CFM}}$. Proof sketch:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,\mathbf{x}\sim p_t}\|v_\theta(\mathbf{x},t) - u_t(\mathbf{x})\|^2$$

Expand and use $\mathbb{E}_{\mathbf{x}\sim p_t}[u_t(\mathbf{x})^\top v_\theta(\mathbf{x},t)] = \mathbb{E}_{\mathbf{z},\mathbf{x}\sim p_t(\cdot|\mathbf{z})}[u_t(\mathbf{x}|\mathbf{z})^\top v_\theta(\mathbf{x},t)]$ (by marginalisation, since $u_t(\mathbf{x}) = \mathbb{E}_{p(\mathbf{z}|\mathbf{x})}[u_t(\mathbf{x}|\mathbf{z})]$ and $\mathbb{E}_{\mathbf{x}\sim p_t}[\cdot] = \mathbb{E}_{\mathbf{z}}\mathbb{E}_{\mathbf{x}\sim p_t(\cdot|\mathbf{z})}[\cdot]$). The squared norm of $u_t(\mathbf{x})$ is a constant

w.r.t. θ, so it drops from the gradient. The $\|u_t(\mathbf{x}|\mathbf{z})\|^2$ term similarly vanishes. This shows $\nabla_\theta \mathcal{L}_{\text{FM}} = \nabla_\theta \mathcal{L}_{\text{CFM}}$.

Gaussian conditional path — full derivation. The simplest conditional path: given data point $\mathbf{x}_1$, define

$$p_t(\mathbf{x}|\mathbf{x}_1) = \mathcal{N}(\mathbf{x};\, t\mathbf{x}_1,\, (1-(1-\sigma_{\min})t)^2\mathbf{I})$$

with $\sigma_{\min} \approx 10^{-3}$. At $t=0$: $p_0(\mathbf{x}|\mathbf{x}_1) = \mathcal{N}(\mathbf{0},\mathbf{I}) = p_0$. At $t=1$: $p_1(\mathbf{x}|\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_1, \sigma_{\min}^2\mathbf{I}) \approx \delta_{\mathbf{x}_1}$.

The conditional vector field that generates this path (i.e. satisfies the conditional continuity equation $\partial_t p_t(\cdot|\mathbf{x}_1) + \nabla\cdot(p_t(\cdot|\mathbf{x}_1)u_t(\cdot|\mathbf{x}_1)) = 0$) is:

$$u_t(\mathbf{x}|\mathbf{x}_1) = \frac{\mathbf{x}_1 - (1-\sigma_{\min})\mathbf{x}}{1-(1-\sigma_{\min})t}$$

Verification: the conditional path can be sampled as $\mathbf{x}_t = t\mathbf{x}_1 + (1-(1-\sigma_{\min})t)\mathbf{x}_0$ with $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0},\mathbf{I})$. The time derivative is:

$$\dot{\mathbf{x}}_t = \mathbf{x}_1 - (1-\sigma_{\min})\mathbf{x}_0 = \frac{\mathbf{x}_1 - (1-\sigma_{\min})\mathbf{x}_t}{1-(1-\sigma_{\min})t} = u_t(\mathbf{x}_t|\mathbf{x}_1)\checkmark$$

Training algorithm. CFM training is simulation-free:

1. Sample $\mathbf{x}_1 \sim p_{\text{data}}$, $t \sim \mathcal{U}[0,1]$, $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0},\mathbf{I})$.
2. Compute $\mathbf{x}_t = t\mathbf{x}_1 + (1-(1-\sigma_{\min})t)\mathbf{x}_0$ and target $u_t = \mathbf{x}_1 - (1-\sigma_{\min})\mathbf{x}_0$.
3. Gradient step: $\theta \leftarrow \theta - \eta\nabla_\theta\|v_\theta(\mathbf{x}_t,t) - u_t\|^2$.

Each step costs exactly one forward pass — identical overhead to DDPM.

Inference. To sample: draw $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ and solve the ODE $d\mathbf{x}/dt = v_\theta(\mathbf{x},t)$ from $t=0$ to $t=1$ using any ODE solver (Euler, Heun, RK45). The straight-line training targets mean that v_θ is approximately constant along trajectories, so Euler's method with 10–50 steps achieves high-quality results.

21.3 Optimal Transport CFM (OT-CFM)

Standard CFM problem. In CFM (Section 21.2), pairs $(\mathbf{x}_0,\mathbf{x}_1)$ are sampled independently: $\mathbf{x}_0 \sim p_0$, $\mathbf{x}_1 \sim p_{\text{data}}$. The linear trajectories $\mathbf{z}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1$ may incrociarsi nel spazio $\mathbb{R}^d$, creando un campo di velocità $\mathbf{u}_t$ non rettilineo e richiede più step di integrazione.

Optimal Transport. OT-CFM (Lipman et al., 2022; Tong et al., 2023) uses the optimal transport coupling $\pi^* \in \Pi(p_0, p_{\text{data}})$ that minimises the quadratic cost:

$$\pi^* = \arg\min_{\pi\in\Pi(p_0,p_{\text{data}})} \mathbb{E}_{(\mathbf{x}_0,\mathbf{x}_1)\sim\pi}\left[\|\mathbf{x}_0 - \mathbf{x}_1\|^2\right]$$

Brenier's theorem. For absolutely continuous p_0, p_1, the optimal OT coupling is determined by a map $T : \mathbb{R}^d \to \mathbb{R}^d$ that is the gradient of a convex function ϕ: $T(\mathbf{x}_0) = \nabla\phi(\mathbf{x}_0)$. OT trajectories are straight lines $\mathbf{z}_t = (1-t)\mathbf{x}_0 + tT(\mathbf{x}_0)$ that *do not cross*: the marginal velocity field is already rectilinear, $\mathbf{u}_t(\mathbf{z}) = \text{const}$ lungo ogni traiettoria.

OT-CFM Objective

$$\mathcal{L}_{\text{OT-CFM}}(\theta) = \mathbb{E}_{t,(\mathbf{x}_0,\mathbf{x}_1)\sim\pi^*}\left[\|\mathbf{v}_\theta((1-t)\mathbf{x}_0 + t\mathbf{x}_1,\, t) - (\mathbf{x}_1 - \mathbf{x}_0)\|^2\right]$$

The only difference from standard CFM is that $(\mathbf{x}_0, \mathbf{x}_1)$ are sampled from the OT coupling π^* instead of the independent coupling.

Minibatch OT. Computing the true OT plan between p_0 and p_1 requires solving a linear program over all joint distributions — $O(n^3)$ cost for n samples. Minibatch OT (Fatras et al., 2021) solves the OT problem on mini-batches of size m:

1. Sample $\{\mathbf{x}_0^{(i)}\}_{i=1}^m \sim p_0$ and $\{\mathbf{x}_1^{(j)}\}_{j=1}^m \sim p_1$.
2. Compute optimal assignment $\sigma^* = \arg\min_\sigma \sum_i \|\mathbf{x}_0^{(i)} - \mathbf{x}_1^{(\sigma(i))}\|^2$ (Hungarian algorithm, $O(m^3)$).
3. Train on paired paths $\mathbf{x}_t = (1-t)\mathbf{x}_0^{(i)} + t\mathbf{x}_1^{(\sigma^*(i))}$.

With $m = 32$–128, minibatch OT significantly improves sample quality and training stability over independent coupling.

Practical results. OT-CFM trains faster and generates higher-quality samples than standard CFM (which uses independent coupling), and significantly outperforms DDPM in terms of NFE (number of function evaluations) to reach a given FID. State-of-the-art text-to-image models (Stable Diffusion 3, Flux) use OT-CFM as their training objective.

Stochastic interpolants. Albergo & Vanden-Eijnden (2023) introduce *stochastic interpolants* as a generalisation of both flow matching and diffusion. An interpolant between $\mathbf{x}_0 \sim p_0$ and $\mathbf{x}_1 \sim p_1$ is:

$$\mathbf{x}_t = \alpha(t)\mathbf{x}_0 + \beta(t)\mathbf{x}_1 + \gamma(t)\square, \quad \square \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

with boundary conditions $\alpha(0) = \beta(1) = 1$, $\alpha(1) = \beta(0) = \gamma(0) = \gamma(1) = 0$. The instantaneous velocity is:

$$\dot{\mathbf{x}}_t = \dot{\alpha}(t)\mathbf{x}_0 + \dot{\beta}(t)\mathbf{x}_1 + \dot{\gamma}(t)\square$$

The optimal learned velocity field $v_\theta(\mathbf{x}, t)$ minimises $\mathbb{E}[\|v_\theta(\mathbf{x}_t, t) - \dot{\mathbf{x}}_t\|^2]$. Setting $\gamma(t) = 0$ recovers flow matching; setting $\alpha(t) = \sqrt{1-t^2}$, $\beta(t) = t$, $\gamma(t) = \epsilon\sqrt{1-t^2}$ recovers an interpolation between a diffusion schedule and a straight-line path. Stochastic interpolants unify all these models under a single regression objective.

Scalability of flow matching. OT-CFM achieves state-of-the-art performance on class-conditional image generation (DiT with FM backbone) with fewer NFEs (Number of Function Evaluations) than DDPM: while DDPM requires $T = 1000$ steps for high-quality samples, OT-CFM achieves comparable quality in 10–50 steps. The straighter trajectories (due to OT coupling) reduce the discretisation error of the ODE solver, enabling fewer integration steps.

Why straight trajectories reduce NFE. The error of Euler's method for an ODE $d\mathbf{x}/dt = v(t)$ over interval $[0, 1]$ with step size $h = 1/N$ is $O(h\int_0^1 \|v'(t)\|dt)$. For a perfectly straight trajectory, $v'(t) = 0$ everywhere, and the error is exactly zero regardless of N. OT-CFM minimises trajectory curvature by finding couplings $(\mathbf{x}_0, \mathbf{x}_1)$ such that the straight-line path $\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1$ corresponds to the true OT geodesic, minimising $\int_0^1 \|v(t)\|^2 dt$ over all flows connecting p_0 to p_1.

Wasserstein-2 distance and OT-CFM. The W_2 distance between p_0 and p_1 is:

$$W_2^2(p_0, p_1) = \inf_{\pi\in\Pi(p_0,p_1)} \mathbb{E}_{(\mathbf{x}_0,\mathbf{x}_1)\sim\pi}\left[\|\mathbf{x}_0 - \mathbf{x}_1\|^2\right]$$

The OT-CFM training loss at optimum satisfies:

$$\min_\theta \mathcal{L}_{\text{OT-CFM}} = W_2^2(p_0, p_1) - \text{const}$$

This provides a PAC-Bayes-style interpretation: OT-CFM minimises the expected squared distance between noise and data samples along optimal straight-line paths.

Minibatch OT convergence. The minibatch OT coupling $\hat{\pi}_m$ converges to the true OT coupling π^* as $m \to \infty$ in the weak topology. For $m = 128$ and Gaussian p_0, p_1 in $d = 100$, minibatch OT achieves $> 90\%$ of the reduction in path crossing relative to independent coupling.

Connection to Schrödinger bridge. The Schrödinger bridge problem (De Bortoli et al., 2021) finds the entropy-regularised OT coupling:

$$\pi^*_\lambda = \arg\min_\pi \mathbb{E}_\pi[\|\mathbf{x}_0 - \mathbf{x}_1\|^2] + \lambda\, D_{\text{KL}}(\pi \,\|\, \pi_{\text{ref}})$$

where $\pi_{\text{ref}} = p_0 \otimes p_1$ (independent coupling). As $\lambda \to 0$, $\pi^*_\lambda \to \pi^*_{\text{OT}}$ (pure OT). Schrödinger bridges produce smoother, more regularised couplings and can be solved iteratively via IPF (Iterative Proportional Fitting). OT-CFM and bridge matching are both special cases of this framework.

22.1 Energy-Based Models

Energy-Based Model (EBM)

An EBM defines a probability distribution implicitly via an energy function $E_\theta : \mathbb{R}^d \to \mathbb{R}$:

$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z(\theta)}, \qquad Z(\theta) = \int \exp(-E_\theta(\mathbf{x}))\, d\mathbf{x}$$

The normalisation constant $Z(\theta)$ (partition function) is intractable in high dimensions. For this reason EBMs cannot be trained with MLE diretta.

MLE for EBM. The log-likelihood gradient is:

$$\begin{aligned}\nabla_\theta \log p_\theta(\mathbf{x}) &= -\nabla_\theta E_\theta(\mathbf{x}) - \nabla_\theta \log Z(\theta) \\ &= -\nabla_\theta E_\theta(\mathbf{x}) + \mathbb{E}_{p_\theta(\tilde{\mathbf{x}})}[\nabla_\theta E_\theta(\tilde{\mathbf{x}})]\end{aligned}$$

The second term requires sampling from p_θ — intractable. Used approaches:

- **Contrastive Divergence (CD)**: campionamento MCMC corto (Hinton, 2002). Bias ma praticamente utile.
- **Stochastic Maximum Likelihood (SML)**: catene MCMC persistenti.
- **Score Matching**: evita $Z(\theta)$ completamente (Sezione 22.2).
- **Noise Contrastive Estimation (NCE)**: trasforma l'MLE in classificazione binaria tra dati reali e rumore.

Contrastive Divergence. Hinton (2002) proposes estimating the second term with k Gibbs/Langevin steps initialised from the real data:

$$\nabla_\theta \log p_\theta(\mathbf{x}) \approx -\nabla_\theta E_\theta(\mathbf{x}) + \nabla_\theta E_\theta(\tilde{\mathbf{x}}^{(k)})$$

where $\tilde{\mathbf{x}}^{(k)}$ is generated with k MCMC steps starting from $\mathbf{x}$. CD-1 ($k = 1$) è praticamente efficace ma teoricamente biased.

EBM sampling with Langevin dynamics. The most common MCMC sampler for continuous EBMs is stochastic Langevin dynamics:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \frac{\alpha}{2}\nabla_{\mathbf{x}} E_\theta(\mathbf{x}^{(t)}) + \sqrt{\alpha}\,\boldsymbol{\eta}^{(t)}, \qquad \boldsymbol{\eta}^{(t)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

In the limit $\alpha \to 0$ and $t \to \infty$ the stationary distribution is p_θ. The term $-\nabla_{\mathbf{x}} E_\theta$ pushes toward low-energy (high-probability) regions; the noise $\sqrt{\alpha}\boldsymbol{\eta}$ prevents collapse to local minima.

Example — EBM as a Unifying Model

Classical machine learning models are special cases of EBMs:

- **Logistic regression**: $E(\mathbf{x}, y) = -y\,\mathbf{w}^\top \mathbf{x}$ (discretised categories).
- **Boltzmann Maltzmann ristretta (RBM)**: $E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^\top \mathbf{W}\mathbf{h} - \mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h}$.
- **Score-based models (Parte 22.2)**: $E_\theta(\mathbf{x})$ con $s_\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} E_\theta(\mathbf{x})$.

Definition: Energy-Based Model (EBM)

$$p_\theta(\mathbf{x}) = \frac{e^{-E_\theta(\mathbf{x})}}{Z(\theta)}, \qquad Z(\theta) = \int e^{-E_\theta(\mathbf{x})}\, d\mathbf{x}$$

where $E_\theta : \mathbb{R}^d \to \mathbb{R}$ is the energy function and $Z(\theta)$ is the partition function.

Problem. $Z(\theta)$ is intractable for high-dimensional $\mathbf{x}$. The gradient of the log-likelihood:

$$\nabla_\theta \log p_\theta(\mathbf{x}) = -\nabla_\theta E_\theta(\mathbf{x}) + \mathbb{E}_{p_\theta}[\nabla_\theta E_\theta(\mathbf{x}')]$$

The second term requires sampling from p_θ, which is expensive (MCMC).

Training EBMs: the log-likelihood gradient. The gradient of the log-likelihood:

$$\nabla_\theta \log p_\theta(\mathbf{x}) = -\nabla_\theta E_\theta(\mathbf{x}) + \mathbb{E}_{p_\theta(\mathbf{x}')}[\nabla_\theta E_\theta(\mathbf{x}')]$$

- *Positive phase* $-\nabla_\theta E_\theta(\mathbf{x})$: decreases the energy of data points (pushes $p_\theta(\mathbf{x})$ up).
- *Negative phase* $+\mathbb{E}_{p_\theta}[\nabla_\theta E_\theta]$: increases the energy of model samples (pushes $p_\theta(\mathbf{x}')$ down).

The challenge: the negative phase requires samples from the current model p_θ — the partition function makes this hard.

Sampling methods for the negative phase.

- *MCMC / Langevin dynamics*: $\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\epsilon}{2}\nabla_{\mathbf{x}} E_\theta(\mathbf{x}_k) + \sqrt{\epsilon}\,\mathbf{z}_k$. Requires many steps; slow.
- *Contrastive divergence* (§22.3): approximate negative phase with k short Markov chains.
- *Replay buffer*: store a buffer of past MCMC samples and reuse them as initial states, reducing mixing time.

Implicit EBMs via score. Instead of specifying E_θ explicitly, *score-based EBMs* (§22.2) learn $s_\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} E_\theta(\mathbf{x})$ directly without computing Z_θ. Sampling proceeds via Langevin dynamics using the learned score.

Example — Ising Model as an EBM

Binary lattice $\mathbf{x} \in \{-1, +1\}^N$, energy $E(\mathbf{x}) = -J \sum_{\langle i,j \rangle} x_i x_j - h \sum_i x_i$.
$J > 0$: ferromagnetic (neighbours prefer alignment). $Z = \sum_{\mathbf{x}} \exp(-\beta E(\mathbf{x}))$ is intractable for large N. Sampling via Gibbs (CD-k) or Metropolis-Hastings. The 2D Ising model is a prototypical EBM used to study phase transitions and is equivalent to a shallow Boltzmann machine.

22.2 Score Matching

The partition function problem. In MLE for EBMs, the gradient w.r.t. θ includes $\mathbb{E}_{p_\theta}[\nabla_\theta E_\theta]$ — intractable when $Z(\theta)$ is unknown. Score matching (Hyvärinen, 2005) evita del tutto $Z(\theta)$.

Score Function e Score Matching

Lo *score* di p_θ è $\mathbf{s}_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} E_\theta(\mathbf{x})$.
L'obiettivo di score matching è minimizzare la distanza attesa tra gli score:

$$J(\theta) = \frac{1}{2} \mathbb{E}_{p_{\text{data}}} [\|\mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})\|^2]$$

Integration by parts formula. The gradient of $\log p_{\text{data}}$ is unknown, but Hyvärinen shows that (integrating by parts under boundary conditions):

$$J(\theta) = \mathbb{E}_{p_{\text{data}}} \left[\sum_i \left(\partial_i s_{\theta,i}(\mathbf{x}) + \frac{1}{2} s_{\theta,i}(\mathbf{x})^2 \right) \right] + \text{const}$$

where $\partial_i s_{\theta,i} = \partial^2 \log p_\theta / \partial x_i^2$. Thisto elimina p_{data} dal gradiente — il calcolo richiede solo valutare $\mathbf{s}_\theta$ e le sue derivate seconde.

Denoising Score Matching (DSM). Vincent (2011) proposes an efficient alternative. Gaussian noise is added: $\tilde{\mathbf{x}} = \mathbf{x} + \sigma \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. L'obiettivo DSM è:

$$J_{\text{DSM}}(\theta; \sigma) = \mathbb{E}_{\mathbf{x}, \boldsymbol{\epsilon}} \left[\left\| \mathbf{s}_\theta(\tilde{\mathbf{x}}) + \frac{\boldsymbol{\epsilon}}{\sigma} \right\|^2 \right]$$

The target is $-\boldsymbol{\epsilon}/\sigma = \nabla_{\tilde{\mathbf{x}}} \log p_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$: the score of the Gaussian noise kernel, which is analytic. This reduces score learning to regression.

Multi-scale score matching. Song & Ermon (2019) propose training a score network $\mathbf{s}_\theta(\mathbf{x}, \sigma)$ on a set of noise levels $\{\sigma_i\}_{i=1}^L$ with $\sigma_1 > \cdots > \sigma_L > 0$:

$$\mathcal{L}(\theta) = \frac{1}{L} \sum_{i=1}^{L} \sigma_i^2 \, \mathbb{E}_{\mathbf{x}, \boldsymbol{\epsilon}} \left[\left\| \mathbf{s}_\theta(\tilde{\mathbf{x}}_i, \sigma_i) + \frac{\boldsymbol{\epsilon}}{\sigma_i} \right\|^2 \right]$$

The weight σ_i^2 balances the scales. This is the precursor of the DDPM: sostituendo lo score con il noise predictor ($\mathbf{s}_\theta = -\boldsymbol{\epsilon}_\theta / \sigma$), si recupera l'Equazione 20.3.

Annealed Langevin sampling. After training $\mathbf{s}_\theta$, generation proceeds by sampling with Langevin dynamics at decreasing noise scales:

1. For $i = 1, \ldots, L$ (scala di rumore decrescente):
2. Inizializza $\mathbf{x}_0$ da $\mathcal{N}(\mathbf{0}, \sigma_1^2 \mathbf{I})$.
3. Esegui T passi: $\mathbf{x}_{t+1} = \mathbf{x}_t + \alpha_i \mathbf{s}_\theta(\mathbf{x}_t, \sigma_i) + \sqrt{2\alpha_i} \boldsymbol{\eta}_t$.

This is *annealed Langevin sampling* (Song & Ermon, 2019), a direct precursor of diffusion models.

The *score function* of a distribution is $s_\theta(\mathbf{x}) \triangleq \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} E_\theta(\mathbf{x})$.

Score matching (Hyvärinen, 2005) avoids $Z(\theta)$ by matching the score directly:

$$\mathcal{L}_{\text{SM}} = \mathbb{E}_{p_{\text{data}}}\left[\frac{1}{2}\left\|s_\theta(\mathbf{x})\right\|^2 + \text{tr}(\nabla_{\mathbf{x}} s_\theta(\mathbf{x}))\right]$$

Integration by parts shows this equals $\frac{1}{2}\mathbb{E}[\|s_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})\|^2] + \text{const}$ — minimising $\mathcal{L}_{\text{SM}}$ is equivalent to minimising the score MSE, without needing p_{data} explicitly.

Denoising Score Matching. Add noise $\mathbf{x}' = \mathbf{x} + \sigma\square$ and match the score of the noised distribution:

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{\mathbf{x},\square}\left[\left\|s_\theta(\mathbf{x}') + \frac{\square}{\sigma}\right\|^2\right]$$

This connects directly to diffusion model training: $\square_\theta = -\sigma s_\theta$.

Integration by parts trick. The original score matching objective:

$$J(\theta) = \frac{1}{2}\mathbb{E}_p[\|s_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{x})\|^2]$$

requires knowing $\nabla \log p$ — unavailable (it's what we're trying to learn). Hyvärinen's trick: expand the squared norm and integrate $\mathbb{E}_p[\nabla \log p \cdot s_\theta]$ by parts:

$$\mathbb{E}_p[\nabla_{\mathbf{x}} \log p(\mathbf{x}) \cdot s_\theta(\mathbf{x})] = -\mathbb{E}_p[\text{tr}(\nabla_{\mathbf{x}} s_\theta(\mathbf{x}))]$$

assuming boundary terms vanish ($p \to 0$ at infinity). Result:

$$J(\theta) = \mathbb{E}_p\left[\text{tr}(\nabla_{\mathbf{x}} s_\theta) + \tfrac{1}{2}\|s_\theta\|^2\right] + \text{const}$$

This is computable! Cost: $O(d^2)$ for the Jacobian trace; Hutchinson's estimator reduces to $O(d)$.

Sliced score matching. Projecting onto random directions $\mathbf{v} \sim p_v$:

$$J_{\text{sliced}}(\theta) = \mathbb{E}_{\mathbf{v},\mathbf{x}}\left[\mathbf{v}^\top \nabla_{\mathbf{x}} s_\theta \mathbf{v} + \tfrac{1}{2}(\mathbf{v}^\top s_\theta)^2\right]$$

Reduces cost to $O(d)$ per sample, enabling training on high-dimensional data (images).

Denoising score matching. Add Gaussian noise $\tilde{\mathbf{x}} = \mathbf{x} + \sigma\mathbf{z}$, $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and train $s_\theta(\tilde{\mathbf{x}}, \sigma) \approx \nabla_{\tilde{\mathbf{x}}} \log p_\sigma(\tilde{\mathbf{x}})$. The optimal score is $-\mathbf{z}/\sigma$. DSM objective:

$$J_{\text{DSM}}(\theta) = \mathbb{E}_{\mathbf{x},\mathbf{z}}\left[\left\|s_\theta(\mathbf{x} + \sigma\mathbf{z}, \sigma) + \frac{\mathbf{z}}{\sigma}\right\|^2\right]$$

This is exactly the DDPM noise prediction loss at a single noise level σ. DSM avoids the Jacobian computation entirely.

22.3 Contrastive Divergence Training

For EBMs $p_\theta(\mathbf{x}) = \exp(-E_\theta(\mathbf{x}))/Z_\theta$, the log-likelihood gradient is:

$$\nabla_\theta \log p_\theta(\mathbf{x}) = -\nabla_\theta E_\theta(\mathbf{x}) + \mathbb{E}_{p_\theta(\mathbf{x}')}[\nabla_\theta E_\theta(\mathbf{x}')]$$

The second term ("negative phase") requires sampling from p_θ — expensive Markov chain Monte Carlo.

Contrastive divergence (CD-k). Replace MCMC samples with k steps of Gibbs sampling starting from data:

1. Initialise chain from data: $\mathbf{x}^{(0)} = \mathbf{x}_{\text{data}}$.
2. Run k MCMC steps: $\mathbf{x}^{(k)} \sim p_\theta(\cdot)$ (approximately).
3. Use $\mathbf{x}^{(k)}$ as the negative sample.

CD-1 (1 step) is surprisingly effective in practice for RBMs.

Persistent CD (PCD). Instead of restarting the Markov chain from data each step, PCD maintains a persistent chain across updates. This reduces the bias introduced by CD-k (which never reaches the model distribution's stationary distribution). The chain "remembers" where it was, gradually exploring p_θ as training proceeds.

Bias of CD-k. CD-k minimises not the KL divergence $\mathrm{KL}(p_{\text{data}} \| p_\theta)$ but an approximation. For $k = 1$, the gradient is biased and can converge to a local minimum different from the MLE solution. In practice, CD-1 works well for RBMs on MNIST but may fail for complex distributions.

Example — CD-1 Update for an RBM

Data batch $\{\mathbf{v}^{(n)}\}_{n=1}^{B}$.

1. **Positive phase:** $\langle v_i h_j \rangle_{\text{data}} = \frac{1}{B} \sum_n v_i^{(n)} \sigma(\mathbf{W}_{:j}^\top \mathbf{v}^{(n)} + c_j)$.
2. **Gibbs step:** sample $\hat{\mathbf{h}}^{(n)} \sim p(\mathbf{h}|\mathbf{v}^{(n)})$, then $\hat{\mathbf{v}}^{(n)} \sim p(\mathbf{v}|\hat{\mathbf{h}}^{(n)})$.
3. **Negative phase:** $\langle v_i h_j \rangle_{\text{model}} = \frac{1}{B} \sum_n \hat{v}_i^{(n)} \sigma(\mathbf{W}_{:j}^\top \hat{\mathbf{v}}^{(n)} + c_j)$.
4. **Update:** $\mathbf{W} \leftarrow \mathbf{W} + \eta(\langle \mathbf{v}\mathbf{h}^\top \rangle_{\text{data}} - \langle \mathbf{v}\mathbf{h}^\top \rangle_{\text{model}})$.

The positive phase pushes up the probability of data configurations; the negative phase pushes down the probability of model "fantasies".

Block Gibbs sampling for RBMs. In an RBM, $p(\mathbf{h}|\mathbf{v}) = \prod_j p(h_j|\mathbf{v})$ and $p(\mathbf{v}|\mathbf{h}) = \prod_i p(v_i|\mathbf{h})$ are fully factored. Each conditional is a Bernoulli:

$$p(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_i W_{ij} v_i\right), \quad p(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_j W_{ij} h_j\right)$$

One step of block Gibbs samples all hidden units simultaneously given visible (and vice versa). CD-k runs k such alternating steps starting from a training example $\mathbf{v}^{(0)}$, producing $\mathbf{v}^{(k)}$:

$$\Delta W_{ij} \propto \mathbb{E}_{\text{data}}[v_i h_j] - \mathbb{E}_{\mathbf{v}^{(k)}}[v_i h_j]$$

where $\mathbb{E}_{\text{data}}[v_i h_j] = v_i^{(0)} \sigma(b_j + W^\top \mathbf{v}^{(0)})_j$ is computed analytically (no sampling needed for the positive phase), and $\mathbb{E}_{\mathbf{v}^{(k)}}[v_i h_j]$ is estimated from the k-step fantasy particle.

Why CD-1 works empirically. Although CD-1 is a biased estimator of the log-likelihood gradient, it captures the first-order discrepancy between the data distribution and the model distribution near the data manifold. For well-behaved unimodal distributions, one step of Gibbs is sufficient to move from the data to the model's nearest mode. The bias becomes problematic for multimodal or highly structured distributions.

Relationship to score matching. For continuous-valued RBMs, CD-1 is equivalent to a first-order approximation of denoising score matching (Hyvärinen, 2007). This connection motivates modern EBM training via DSM (§22.2), which is unbiased and handles complex distributions better than CD.

Stochastic Gradient Langevin Dynamics (SGLD) for EBMs. Du & Mordatch (2019) train

EBMs by running K steps of SGLD to produce negative samples:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \frac{\alpha}{2}\nabla_{\mathbf{x}} E_\theta(\mathbf{x}^{(k)}) + \sqrt{\alpha}\,\mathbf{z}^{(k)}, \quad \mathbf{z}^{(k)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

With step size $\alpha \to 0$ and $K \to \infty$, SGLD converges to the Gibbs distribution $p_\theta(\mathbf{x}) \propto \exp(-E_\theta(\mathbf{x}))$. In practice, $K = 60$ Langevin steps with a replay buffer of 10^4 persistent chains achieves competitive image generation (FID ≈ 40 on CIFAR-10).

Replay buffer. Maintaining a buffer of "fantasy" samples from previous model versions dramatically improves EBM training stability. Buffer samples are initialised from data; as training proceeds, they drift toward the current model's distribution. At each step:

1. With probability $p_{\text{reinit}} = 0.05$, reinitialise a buffer sample from $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$.
2. Otherwise, continue the Langevin chain from the stored sample.

This prevents mode collapse (chains getting stuck at a single mode) while maintaining the benefits of PCD (persistent chains).

Convergence of Langevin dynamics. Under log-Sobolev inequality with constant $\rho > 0$ (a smoothness condition on p_θ), the KL divergence between the Langevin chain distribution μ_t and the target p_θ satisfies:

$$D_{\text{KL}}(\mu_t \,\|\, p_\theta) \leq e^{-2\rho t} D_{\text{KL}}(\mu_0 \,\|\, p_\theta)$$

Convergence is geometric in t (continuous time). For discrete Langevin with step size α, the mixing time is $O(\rho^{-1} \log D_{\text{KL}}(\mu_0 \,\|\, p_\theta)/\epsilon)$ steps to reach ϵ-error. In high dimensions, ρ typically scales as $O(1/d)$, meaning $O(d)$ steps are needed — expensive for large d.

Parallel tempering for multimodal distributions. MCMC chains mix poorly when p_θ has well-separated modes. Parallel tempering runs L chains at temperatures $T_1 < T_2 < \cdots < T_L$, where chain l targets p_θ^{1/T_l}. High-temperature chains mix easily; swaps between adjacent temperature levels occasionally transfer samples from well-mixed hot chains to cooler ones. For EBM training, this can be combined with PCD to improve coverage of the model distribution.

22.4 Noise Contrastive Estimation (NCE)

NCE (Gutmann & Hyvärinen 2010) trains the EBM to distinguish data from noise. Define a noise distribution $q(\mathbf{x})$ and train a binary classifier:

$$p(\text{data}|\mathbf{x}) = \frac{p_\theta(\mathbf{x})}{p_\theta(\mathbf{x}) + kq(\mathbf{x})}, \qquad p(\text{noise}|\mathbf{x}) = \frac{kq(\mathbf{x})}{p_\theta(\mathbf{x}) + kq(\mathbf{x})}$$

where k noise samples are drawn per data sample. The model p_θ is parameterised as $\exp(-E_\theta(\mathbf{x}) + c)$ where c is a learned normalisation constant — avoiding the need to compute Z_θ.

Connection to contrastive learning. InfoNCE (Part 31) is a multi-class extension of NCE and is used in CLIP, SimCLR, and other self-supervised methods.

PART 23

Deep Gaussian Processes

A Gaussian Process places a prior over functions. Deep GPs compose multiple GP layers, inheriting the expressive power of deep networks while maintaining Bayesian uncertainty quantification.

Theoretical justification. NCE reduces density estimation to binary classification. The log-likelihood of the classifier:

$$\ell(\theta) = \mathbb{E}_{p_{\text{data}}}\left[\log \frac{p_\theta(\mathbf{x})}{p_\theta(\mathbf{x}) + kq(\mathbf{x})}\right] + k\,\mathbb{E}_q\left[\log \frac{kq(\mathbf{x})}{p_\theta(\mathbf{x}) + kq(\mathbf{x})}\right]$$

At the optimum (as $|\mathcal{D}| \to \infty$): $p^*_\theta(\mathbf{x}) = p_{\text{data}}(\mathbf{x})$. The learned constant c converges to $\log Z_\theta$, i.e. the model implicitly estimates the partition function.

Choice of noise distribution. q should be easy to sample from and close to p_{data} (for low-variance estimates). Common choices:

- Gaussian with data mean and variance.
- Unigram distribution over discrete tokens (used in word2vec, GPT language model heads).
- Uniform distribution (simpler but less efficient).

InfoNCE and contrastive learning. InfoNCE (van den Oord et al., 2018) extends NCE to the multi-class setting with K noise samples per data point:

$$\ell_{\text{InfoNCE}} = -\mathbb{E}\left[\log \frac{\exp(f(\mathbf{x}, \mathbf{c})/\tau)}{\exp(f(\mathbf{x}, \mathbf{c})/\tau) + \sum_{k=1}^{K-1} \exp(f(\mathbf{x}, \mathbf{c}_k)/\tau)}\right]$$

This is a lower bound on mutual information $I(\mathbf{x}; \mathbf{c}) \geq \log K - \ell_{\text{InfoNCE}}$. InfoNCE is the objective of SimCLR, MoCo, CLIP, and most modern self-supervised learning frameworks.

NCE as logistic regression. NCE trains the model to distinguish real data from noise. Given k noise samples per real sample, the NCE objective is:

$$\mathcal{L}_{\text{NCE}}(\theta) = \mathbb{E}_{\mathbf{x}\sim p_d}[\log \sigma(h_\theta(\mathbf{x}))] + k\,\mathbb{E}_{\mathbf{y}\sim q}[\log(1 - \sigma(h_\theta(\mathbf{y})))]$$

where $h_\theta(\mathbf{x}) = \log p_\theta(\mathbf{x}) - \log q(\mathbf{x}) + \log k$ is the log-odds ratio. At the optimum, $\sigma(h_\theta(\mathbf{x})) = kp_d(\mathbf{x})/(kp_d(\mathbf{x}) + q(\mathbf{x}))$, which gives $p_\theta = p_d$. The normalisation constant Z_θ can be treated as a free parameter or fixed; treating it as a parameter and initialising $\log Z \approx 1$ works well in practice.

InfoNCE and contrastive representation learning. InfoNCE (van den Oord et al., 2018) extends NCE to mutual information maximisation. Given positive pairs $(\mathbf{x}, \mathbf{c})$ and $N - 1$ negative contexts $\{\mathbf{c}_j\}$:

$$\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}\left[\log \frac{f(\mathbf{x}, \mathbf{c})}{\sum_{j=1}^{N} f(\mathbf{x}, \mathbf{c}_j)}\right], \quad f(\mathbf{x}, \mathbf{c}) = \exp(g(\mathbf{x})^\top h(\mathbf{c})/\tau)$$

Minimising $\mathcal{L}_{\text{InfoNCE}}$ maximises a lower bound on $I(\mathbf{x}; \mathbf{c})$: $I(\mathbf{x}; \mathbf{c}) \geq \log N - \mathcal{L}_{\text{InfoNCE}}$. This bound tightens as N increases — the key role of large batch sizes in SimCLR, MoCo, and CLIP (§31.3).

Noise distribution choice. The efficiency of NCE depends critically on the noise distribution q:

- *Uniform noise:* simple but wastes samples on regions far from the data manifold.
- *Unigram distribution* (word frequencies): standard in word2vec, where $q(w) \propto f(w)^{3/4}$ provides a practical heuristic.
- *Model-based noise:* $q = p_\theta^{\text{prev}}$ (previous model iterate). Used in self-play and curriculum learning but requires recomputing q periodically.

For high-dimensional data, as $k \to \infty$, NCE converges to maximum likelihood — the bias from finite noise vanishes.

Example — NCE vs MLE for language models

In a vocabulary of size $V = 50{,}000$, the softmax normalisation costs $O(V)$ per token. NCE replaces this with $k \ll V$ noise comparisons (typically $k = 64$–256), achieving $O(k)$ per update. On a transformer LM, NCE training is $\approx 100\times$ cheaper per step than full softmax MLE, with only a $\sim 1\%$ perplexity penalty at $k = 256$.

23.1 Gaussian Processes

Processo Gaussiano

A *Gaussian process* (GP) is a distribution over functions $f : \mathcal{X} \to \mathbb{R}$ such that any finite subset of values $f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)$ has a joint Gaussian distribuzione gaussiana congiunta:

$$(f(\mathbf{x}_1), \dots, f(\mathbf{x}_n))^\top \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$$

where $\mu_i = m(\mathbf{x}_i)$ and $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ are the *funzione media* e la *funzione kernel* (covarianza).

Common kernel functions.

- **Squared Exponential (SE)**: $k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\ell^2}\right)$. Funzioni infinitamente differenziabili.
- **Matérn-5/2**: $k = \sigma^2(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2})\exp(-\frac{\sqrt{5}r}{\ell})$. Funzioni 2 volte differenziabili, più realistiche per dati fisici.
- **Rational Quadratic**: $k = \sigma^2(1 + \frac{r^2}{2\alpha\ell^2})^{-\alpha}$, mix di SE a scale diverse.
- **Periodic**: $k = \exp(-\frac{2\sin^2(\pi|x-x'|/p)}{\ell^2})$.

Gaussian Process Regression (GPR). Given $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $y_i = f(\mathbf{x}_i) + \epsilon_i$, $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$:

$$p(f_*|\mathbf{x}_*, \mathcal{D}) = \mathcal{N}(f_*;\, \mu_{*|\mathcal{D}},\, \sigma^2_{*|\mathcal{D}})$$

con:

$$\mu_{*|\mathcal{D}} = m(\mathbf{x}_*) + \mathbf{k}_*^\top(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}(\mathbf{y} - \mathbf{m})$$
$$\sigma^2_{*|\mathcal{D}} = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_*$$

where $\mathbf{k}_* = [k(\mathbf{x}_*, \mathbf{x}_i)]_i$ and $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{ij}$.

Hyperparameter optimisation. The kernel parameters (σ, ℓ, σ_n) are optimised by maximis-

ing the log-marginal-likelihood:

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^\top(\mathbf{K}+\sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}+\sigma_n^2\mathbf{I}| - \frac{n}{2}\log(2\pi)$$

Computational cost: $O(n^3)$ for matrix inversion. For $n > 10^4$, used approacheso sparse GP (inducing points) o random features.

GP as a prior over neural networks (NNGP). Neal (1994) showed that an infinite-width neural network with i.i.d. weights converges to a GP. Lee et al. (2018) and Yang (2019) extended questo risultato a reti profonde (Neural Network Gaussian Process, NNGP), stabilendo la connessione teorica tra deep learning e processi gaussiani. Il kernel NNGP per un layer fully connected è definito ricorsivamente: $K^{(0)}(\mathbf{x},\mathbf{x}') = \mathbf{x}^\top\mathbf{x}'/d + b^2$ e $K^{(l)} = \sigma_w^2\,\mathbb{E}_{(u,v)\sim\mathcal{N}(0,\mathbf{K}^{(l-1)})}[\phi(u)\phi(v)] + \sigma_b^2$.

Example — GP for Bayesian optimisation

In Bayesian Optimisation (BO) for hyperparameter tuning:

1. Model $f(\boldsymbol{\lambda})$ = performance of rete con iperparametri $\boldsymbol{\lambda}$ come GP.
2. Seleziona il prossimo punto da valutare massimizzando una *acquisition function*: Expected Improvement $\text{EI}(\boldsymbol{\lambda}) = \mathbb{E}[\max(f(\boldsymbol{\lambda}) - f^*, 0)]$ oppure Upper Confidence Bound $\text{UCB}(\boldsymbol{\lambda}) = \mu(\boldsymbol{\lambda}) + \kappa\sigma(\boldsymbol{\lambda})$.
3. Aggiorna il GP con il nuovo punto e ripeti.

BO con GP richiede tipicamente $O(n^2)$ valutazioni in meno rispetto a grid search per trovare un iperparametro ottimale.

Definition: Gaussian Process

A Gaussian Process is a collection of random variables, any finite subset of which is jointly Gaussian. A GP is fully specified by:

- Mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$
- Covariance (kernel) function $k(\mathbf{x},\mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$

We write $f \sim \mathcal{GP}(m, k)$.

GP regression. Given observations $\mathbf{y} = f(\mathbf{X}) + \square$, $\square \sim \mathcal{N}(\mathbf{0}, \sigma_n^2\mathbf{I})$, the posterior is also a GP with:

$$m^*(\mathbf{x}) = m(\mathbf{x}) + \mathbf{k}(\mathbf{x},\mathbf{X})\left(\mathbf{K}+\sigma_n^2\mathbf{I}\right)^{-1}(\mathbf{y} - m(\mathbf{X}))$$

$$k^*(\mathbf{x},\mathbf{x}') = k(\mathbf{x},\mathbf{x}') - \mathbf{k}(\mathbf{x},\mathbf{X})\left(\mathbf{K}+\sigma_n^2\mathbf{I}\right)^{-1}\mathbf{k}(\mathbf{X},\mathbf{x}')$$

where $\mathbf{K}_{ij} = k(\mathbf{x}_i,\mathbf{x}_j)$.

Cost. $O(N^3)$ for matrix inversion — intractable for large N. Sparse GP approximations (inducing points) reduce this to $O(NM^2)$ with $M \ll N$ inducing points.

Marginalisation consistency. A GP prior is consistent: any finite subset of outputs $(f(\mathbf{x}_1), \dots, f(\mathbf{x}_n))$ follows a multivariate Gaussian. By Kolmogorov's extension theorem, this is sufficient to define a valid stochastic process.

GP regression (exact). Given noise-free observations $\mathbf{y} = \mathbf{f} + \epsilon$, $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma_n^2\mathbf{I})$, the GP posterior at test point $\mathbf{x}_*$:

$$f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y} \sim \mathcal{N}(\mu_*, \sigma_*^2)$$

$$\mu_* = k(\mathbf{x}_*,\mathbf{X})\left[K(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I}\right]^{-1}\mathbf{y}$$

$$\sigma_*^2 = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X}) \left[K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right]^{-1} k(\mathbf{X}, \mathbf{x}_*)$$

Cost: $O(n^3)$ for the matrix inversion (dominant); $O(n^2)$ storage. Intractable for $n > 10^4$.

Sparse GP approximations. Introduce $m \ll n$ inducing points $\mathbf{Z}$ (locations chosen to summarise the data):

- *Subset of regressors (SoR):* $k_{\text{approx}}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{Z}) K(\mathbf{Z}, \mathbf{Z})^{-1} k(\mathbf{Z}, \mathbf{x}')$. Cost: $O(nm^2)$.
- *Sparse variational GP (SVGP, Titsias 2009):* variational posterior $q(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{u}) q(\mathbf{u})\, d\mathbf{u}$ with Gaussian $q(\mathbf{u})$. ELBO tractable in $O(nm^2)$. Mini-batching possible: SVGP scales to $n = 10^6$.

Hyperparameter learning. GP hyperparameters (ℓ, output scale σ_f^2, noise σ_n^2) are optimised by maximising the log marginal likelihood:

$$\log p(\mathbf{y}|\mathbf{X}, \theta) = -\frac{1}{2}\mathbf{y}^\top K_\theta^{-1}\mathbf{y} - \frac{1}{2}\log|K_\theta| - \frac{n}{2}\log 2\pi$$

Three terms: data fit, complexity penalty (Occam's razor), constant. Gradient-based optimisation via L-BFGS or Adam.

23.2 Deep Gaussian Processes

Motivazione. I GP standard sono limitati da: (a) kernel fisso con poche lunghezze di scala; (b) assunzione di stazionarietà; (c) scarsa capacità per funzioni non stazionarie o gerarchiche. I Deep Gaussian Processes (DGP, Damianou & Lawrence, 2013) compongono GP come le reti neurali compongono trasformazioni:

$$\mathbf{h}^{(1)} = f^{(1)}(\mathbf{x}), \quad \mathbf{h}^{(l)} = f^{(l)}(\mathbf{h}^{(l-1)}), \quad \mathbf{y} = f^{(L)}(\mathbf{h}^{(L-1)})$$

where each $f^{(l)} \sim \mathcal{GP}(0, k^{(l)})$.

Variational inference for DGPs. The joint posterior $p(F|\mathbf{y})$ over all layers is intractable. A variational approximation with inducing points $\{Z^{(l)}, U^{(l)}\}$ per ogni strato:

$$q(\{F^{(l)}\}) = \prod_{l=1}^{L} q(F^{(l)}|U^{(l)}) q(U^{(l)})$$

The ELBO factorises over layers and can be optimised with stochastic gradistico.

Structural properties.

- **Effective kernel**: composing GPs with SE kernels produces highly flexible, non-stationary kernels. Depth permette di apprendere trasformazioni di features.
- **Incertezza epistémica**: a differenza delle reti neurali standard, i DGP forniscono barre d'errore calibrate sull'incertezza (sia aleatoria che epistemica).
- **Connessione con reti neurali**: nel limite di larghezza infinita dei layer hidden del DGP, si recupera un processo gaussiano profondo con kernel ricorsivi (kernel NNGP, Sezione 23.1).

Limitazioni computazionali. L'inferenza variazionale per DGP con n dati, L layer e M inducing points per layer ha costo $O(NM^2L)$ per iterazione. Per $M \sim 100$ e $L \sim 5$ questo è fattibile ma lento per $N > 10^5$. Alternative: random feature approximations, decoupled sampling, o DGP sparsi con kernel Matérn.

Example — DGP vs. Bayesian Neural Networks

- **DGP**: explicit functional prior via kernel; variational inference with guarantees; interpretability of inducing points.
- **BNN** (Bayesian Neural Network): prior sui pesi; inferenza più flessibile ma meno calibrata; scalabilità con KFAC o last-layer Laplace.

In pratica: DGP preferiti per tabular data con $n < 10^4$ e necessità di calibrazione; BNN preferiti per immagini e NLP con architetture standard.

A Deep GP composes L GP layers:

$$\mathbf{h}_1 = f_1(\mathbf{x}) + \square_1, \quad \mathbf{h}_l = f_l(\mathbf{h}_{l-1}) + \square_l, \quad \mathbf{y} = f_L(\mathbf{h}_{L-1}) + \square_L$$

where each $f_l \sim \mathcal{GP}(0, k_l)$.

Intractability. The marginal $p(\mathbf{y} \mid \mathbf{x})$ requires integrating over all intermediate $\mathbf{h}_l$, which is not analytically tractable. Inference requires doubly stochastic variational inference (Salimbeni & Deisenroth, 2017): a variational posterior over inducing points at each layer, with the reparameterisation trick propagating gradients.

Example — Deep GP intuition

A standard GP is a weighted sum of kernel functions — a single-layer nonlinear model. A Deep GP with two layers composes two such models: the second GP takes as input a random feature representation, not the raw input. This allows the covariance structure to adapt over the input space (non-stationary kernels emerge from stationary ones), giving the expressiveness of a deep network while preserving calibrated uncertainty.

Doubly stochastic VI for DGPs. The ELBO for a L-layer DGP:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q(\mathbf{f}^{(1:L)})}\big[\log p(\mathbf{y}|\mathbf{f}^{(L)})\big] - \sum_{l=1}^{L} D_{\text{KL}}\big(q(\mathbf{u}^{(l)}) \,\|\, p(\mathbf{u}^{(l)})\big)$$

where $\mathbf{u}^{(l)}$ are inducing points at each layer. Inference is doubly stochastic: (1) mini-batch over data, (2) Monte Carlo samples of the path $\mathbf{f}^{(1)} \to \cdots \to \mathbf{f}^{(L)}$. Backpropagation through samples uses the reparameterisation trick (as in VAEs).

Limitations and remedies. A key failure mode: intermediate layers can collapse to a trivial mapping $\mathbf{f}^{(l)}(\mathbf{x}) \approx$ const if the KL penalty dominates. Remedies:

- Initialise lengthscales and output variance of intermediate layers to propagate input variance.
- Use "identity" mean functions $m^{(l)}(\mathbf{x}) = \mathbf{x}$ (residual connection analogue for GPs).
- Orthogonal inter-domain inducing features to reduce posterior collapse.

Relation to neural networks. Neal (1995) showed that a 1-hidden-layer BNN with $H \to \infty$ hidden units converges in distribution to a GP. For deep networks, this limit gives a *deep GP* — but the limiting GP kernel depends on the depth and activation function via the arc-cosine kernel (Cho & Saul, 2009).

Doubly stochastic variational inference for DGPs. Exact inference in a DGP requires integrating over all intermediate GP layers, which is analytically intractable. Salimbeni & Deisen-

roth (2017) propose *doubly stochastic* variational inference:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q(\mathbf{F}^1,\ldots,\mathbf{F}^L)}[\log p(\mathbf{y}|\mathbf{F}^L)] - \sum_{\ell=1}^{L} D_{\text{KL}}(q(\mathbf{U}^\ell) \,\|\, p(\mathbf{U}^\ell))$$

where $\mathbf{U}^\ell$ are the inducing outputs at layer ℓ. The expectation is estimated by sampling: first sample $\mathbf{U}^\ell$ from $q(\mathbf{U}^\ell)$, then sample the GP output given the inducing points, propagating uncertainty layer by layer. This "doubly stochastic" approach avoids the single-sample collapse that plagued earlier DGP approximations.

DGP vs. BNN vs. deep kernel learning.

- *DGP:* places GP priors at each layer. Well-calibrated uncertainty, but training is slow ($O(NM^2L)$). Good for small-to-medium datasets where uncertainty matters.
- *BNN (Bayesian neural network):* places priors on weights. Full flexibility but notoriously difficult to train (MCMC or variational with reparameterisation).
- *Deep kernel learning (DKL, Wilson et al., 2016):* uses a neural network $\phi_\theta(\mathbf{x})$ as a feature extractor, then applies a GP in the feature space: $k(\mathbf{x}, \mathbf{x}') = k_{\text{base}}(\phi_\theta(\mathbf{x}), \phi_\theta(\mathbf{x}'))$. Combines NN flexibility with GP calibration. Trainable end-to-end via marginal likelihood.

Pathologies of deep GP priors. As the number of layers L increases, the DGP prior concentrates around zero: repeated composition of zero-mean GPs collapses toward a degenerate prior (Neal, 1996; Duvenaud et al., 2014). Remedies include:

- Adding skip connections: $\mathbf{f}^\ell(\mathbf{x}) = \text{GP}^\ell(\mathbf{x}) + \mathbf{x}$.
- Initialising with near-linear kernels (e.g. linear + small RBF).
- Constraining the kernel hyperparameters to preserve signal-to-noise ratio across layers.

23.3 GP Regression: Full Derivation

Given training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with $y_i = f(\mathbf{x}_i) + \epsilon_i$, $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$:

Prior. $f \sim \mathcal{GP}(0, k(\cdot,\cdot))$ with kernel k.

Joint distribution.

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K} + \sigma_n^2 \mathbf{I} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**} \end{pmatrix}\right)$$

where $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, $(K_*)_{ij} = k(\mathbf{x}_i, \mathbf{x}_j^*)$, $(K_{**})_{ij} = k(\mathbf{x}_i^*, \mathbf{x}_j^*)$.

Posterior (conditional distribution).

$$\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*))$$

$$\bar{\mathbf{f}}_* = \mathbf{K}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$$

$$\text{cov}(\mathbf{f}_*) = \mathbf{K}_{**} - \mathbf{K}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{K}_*$$

Derivation. Apply the conditional multivariate Gaussian formula: if $\begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} \sim \mathcal{N}\left(\square, \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}\right)$, then $\mathbf{b}|\mathbf{a} \sim \mathcal{N}(\square_b + \Sigma_{ba}\Sigma_{aa}^{-1}(\mathbf{a} - \square_a),\ \Sigma_{bb} - \Sigma_{ba}\Sigma_{aa}^{-1}\Sigma_{ab})$.

Computational cost. Matrix inversion $(\mathbf{K}+\sigma_n^2\mathbf{I})^{-1}$: $O(N^3)$. Prediction: $O(N^2)$ per test point. Approximate GPs (sparse GPs, inducing points) reduce this to $O(NM^2)$ with $M \ll N$ inducing points.

Full posterior derivation. Given observations $\mathbf{y} = \mathbf{f} + \square$ with $\square \sim \mathcal{N}(\mathbf{0}, \sigma_n^2\mathbf{I})$, the joint distribution over observations and test function values $\mathbf{f}_*$ at X_* is:

$$\begin{pmatrix}\mathbf{y}\\ \mathbf{f}_*\end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix}K(X,X)+\sigma_n^2\mathbf{I} & K(X,X_*)\\ K(X_*,X) & K(X_*,X_*)\end{pmatrix}\right)$$

By the Schur complement formula for conditional Gaussians, the posterior is:

$$\mathbf{f}_*|\mathbf{y}, X, X_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, \operatorname{cov}(\mathbf{f}_*))$$

with:

$$\bar{\mathbf{f}}_* = K(X_*,X)\left[K(X,X)+\sigma_n^2\mathbf{I}\right]^{-1}\mathbf{y}$$

$$\operatorname{cov}(\mathbf{f}_*) = K(X_*,X_*) - K(X_*,X)\left[K(X,X)+\sigma_n^2\mathbf{I}\right]^{-1}K(X,X_*)$$

Marginal likelihood and hyperparameter learning. The log marginal likelihood (integrating out $\mathbf{f}$) is:

$$\log p(\mathbf{y}|X,\theta) = -\frac{1}{2}\mathbf{y}^\top(K_\theta+\sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|K_\theta+\sigma_n^2\mathbf{I}| - \frac{n}{2}\log 2\pi$$

The three terms balance data fit, complexity penalty (log-determinant), and a constant. Hyperparameters θ (kernel lengthscale, output variance, noise σ_n^2) are learned by gradient ascent on $\log p(\mathbf{y}|X,\theta)$. This is a Type-II maximum likelihood (empirical Bayes) procedure — no cross-validation needed.

Computational cost. Naive GP regression costs $O(n^3)$ for the Cholesky decomposition of the $n \times n$ kernel matrix, plus $O(n^2)$ storage. For $n > 10^4$, sparse approximations (inducing points) are required. The *Nyström approximation* selects $m \ll n$ inducing points Z and approximates $K(X,X) \approx K(X,Z)K(Z,Z)^{-1}K(Z,X)$, reducing cost to $O(nm^2)$.

Predictive uncertainty. At a test point $\mathbf{x}_*$, the predictive mean $\bar{f}_*$ is a weighted sum of kernel similarities to training points: $\bar{f}_* = \sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}_*)$ where $\square = (K+\sigma_n^2\mathbf{I})^{-1}\mathbf{y}$. The predictive variance $\operatorname{cov}(f_*, f_*) = k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\top(K+\sigma_n^2\mathbf{I})^{-1}\mathbf{k}_*$ decreases monotonically as $\mathbf{x}_*$ approaches training points — the GP becomes more confident near observed data, and reverts to the prior variance $k(\mathbf{x}_*,\mathbf{x}_*)$ far from data.

Example — GP regression on 1D data

With $n = 10$ noisy observations from $f(x) = \sin(x)$, the SE kernel with lengthscale $\ell = 1$ and $\sigma_n = 0.1$: the posterior mean tracks the true function, the posterior variance inflates between observations and at the boundaries, and ± 2 standard deviation bands contain $\approx 95\%$ of future observations. Optimising the marginal likelihood over $\ell \in \{0.1, \ldots, 10\}$ recovers $\hat{\ell} \approx 1$ automatically.

23.4 Kernel Functions and Gaussian Process Priors

The kernel function $k(\mathbf{x}, \mathbf{x}')$ encodes prior beliefs about the function's smoothness and structure.

Kernel	Formula	Prior belief
SE (RBF)	$\exp(-\|\mathbf{x}-\mathbf{x}'\|^2/2\ell^2)$	Infinitely differentiable, smooth
Matérn $\frac{3}{2}$	$(1+\sqrt{3}r/\ell)\exp(-\sqrt{3}r/\ell)$	Once differentiable
Linear	$\mathbf{x}^\top\mathbf{x}'+c$	Affine functions
Periodic	$\exp(-2\sin^2(\pi\|x-x'\|/p)/\ell^2)$	Periodic patterns

Kernels can be combined: $k_1 + k_2$ (additive); $k_1 \cdot k_2$ (interaction); $k(\mathbf{x},\mathbf{x}') = k_1(x_1,x_1') \cdot k_2(x_2,x_2')$ (product of independent dimensions).

PART 24

Putting It All Together: The Generative Modelling Landscape

We compare VAEs, GANs, normalising flows, diffusion models, and flow matching along the axes that matter: likelihood tractability, sample quality, inference speed, and theoretical guarantees.

Mercer's theorem. A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a valid kernel (positive definite) iff it can be expressed as $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}$ for some feature map $\phi : \mathcal{X} \to \mathcal{H}$ into a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$. Equivalently, the Gram matrix $[K]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ is positive semidefinite for any finite set of points.

Standard kernels.

Kernel	Formula	Hyperparameters
RBF (SE)	$\exp(-\lVert\mathbf{x}-\mathbf{x}'\rVert^2/2\ell^2)$	length-scale ℓ
Matérn $\nu = 3/2$	$(1+\frac{\sqrt{3}r}{\ell})\exp(-\frac{\sqrt{3}r}{\ell})$	ℓ; $r = \lVert\mathbf{x}-\mathbf{x}'\rVert$
Periodic	$\exp(-\frac{2\sin^2(\pi r/p)}{\ell^2})$	period p, length-scale ℓ
Linear	$\sigma_b^2 + \sigma_v^2 \mathbf{x}^\top \mathbf{x}'$	σ_b, σ_v
Polynomial	$(\sigma^2 + \mathbf{x}^\top \mathbf{x}')^d$	σ, d

Kernel composition. New kernels from old: $k_1 + k_2$, $k_1 \cdot k_2$, $f(\mathbf{x}) \cdot k(\mathbf{x}, \mathbf{x}') \cdot f(\mathbf{x}')$ are all valid kernels. This enables designing structured priors: e.g. $k_{\text{SE}} \cdot k_{\text{periodic}}$ for quasi-periodic functions; $k_{\text{linear}} + k_{\text{SE}}$ for trending functions with local variation.

Neural tangent kernel (NTK). For an infinitely-wide neural network trained with gradient descent, the kernel between inputs $\mathbf{x}$ and $\mathbf{x}'$ at initialisation is the NTK:

$$k_{\text{NTK}}(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{\theta \sim \text{init}}\left[\nabla_\theta f(\mathbf{x}; \theta)^\top \nabla_\theta f(\mathbf{x}'; \theta)\right]$$

In the infinite-width limit, training with gradient descent is equivalent to kernel regression with the NTK. This provides a tractable analysis of neural network training and generalisation.

Spectral view of kernels. By Bochner's theorem, a stationary kernel $k(\mathbf{x} - \mathbf{x}')$ is positive definite iff it is the Fourier transform of a non-negative measure Λ:

$$k(\mathbf{r}) = \int e^{i \square^\top \mathbf{r}} \, \Lambda(d\square)$$

For the SE kernel: $\Lambda = \mathcal{N}(\mathbf{0}, (2\ell^2)^{-1}\mathbf{I})$ — a Gaussian spectral density. For Matérn$_\nu$: Λ is a Student-t distribution. This spectral representation allows constructing kernels by specifying desired frequency content.

Random Fourier features (Rahimi & Recht, 2007). Approximate $k(\mathbf{x}, \mathbf{x}')$ with a finite-dimensional feature map:

$$k(\mathbf{x}, \mathbf{x}') \approx \phi(\mathbf{x})^\top \phi(\mathbf{x}'), \quad \phi(\mathbf{x}) = \sqrt{2/D}\left[\cos(\square_1^\top \mathbf{x} + b_1), \ldots, \cos(\square_D^\top \mathbf{x} + b_D)\right]$$

where $\square_j \sim \Lambda$ and $b_j \sim \mathcal{U}[0, 2\pi]$. This reduces GP inference from $O(n^3)$ to $O(nD^2 + D^3)$ for $D \ll n$ features, enabling approximate GPs on millions of data points.

Automatic Relevance Determination (ARD). Replace the isotropic length-scale ℓ with a vector $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_d)$, one per input dimension:

$$k_{\text{ARD}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\sum_{i=1}^{d} \frac{(x_i - x_i')^2}{\ell_i^2}\right)$$

Dimensions with large ℓ_i contribute less to the kernel (the GP "ignores" them). After marginal likelihood optimisation, $\ell_i \to \infty$ for irrelevant features, performing automatic feature selection. This is the kernel analogue of L_2 regularisation and sparsity.

Deep kernels. Wilson et al. (2016) propose learning a feature map $\phi_\theta : \mathbb{R}^d \to \mathbb{R}^D$ (a neural network) and applying a base kernel in feature space:

$$k_\theta(\mathbf{x}, \mathbf{x}') = k_{\text{base}}(\phi_\theta(\mathbf{x}),\, \phi_\theta(\mathbf{x}'))$$

The resulting deep kernel GP is trained end-to-end by maximising the marginal likelihood w.r.t. $(\theta, k\text{-hyperparameters})$. This combines the representational power of neural networks with the calibrated uncertainty of GPs.

Hyperparameter optimisation via marginal likelihood. Given observations $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$, the (log) marginal likelihood:

$$\log p(\mathbf{y}|\mathbf{X}, \square) = -\frac{1}{2}\mathbf{y}^\top (K + \sigma^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|K + \sigma^2\mathbf{I}| - \frac{n}{2}\log 2\pi$$

is optimised w.r.t. kernel hyperparameters $\square$ (length-scale ℓ, signal variance σ_f^2, noise σ^2) by gradient ascent. The gradient $\partial \log p / \partial \theta_j = \frac{1}{2}\text{tr}\big[(\alpha\alpha^\top - K^{-1})\partial K/\partial\theta_j\big]$ where $\alpha = K^{-1}\mathbf{y}$ costs $O(n^2)$ per hyperparameter after the $O(n^3)$ Cholesky factorisation.

Example — Kernel selection on synthetic data

Data: $y = \sin(2x) + 0.3\cos(10x) + \epsilon$, $\epsilon \sim \mathcal{N}(0, 0.01)$, $n = 50$ points.

- SE kernel ($\ell = 0.2$): fits the low-frequency trend but misses high-frequency oscillations.
- Matérn$_{5/2}$ ($\ell = 0.15$): better interpolation near data; wider uncertainty far from data.
- $k_{\text{SE}} + k_{\text{periodic}}$ (period $p = \pi/5$): correctly captures both components; NLML $= 12.3$ vs 18.7 for pure SE.

This illustrates how kernel composition encodes domain knowledge.

24.1 Comparison of Generative Frameworks

Quattro assi di valutazione. I modelli generativi studiati in questo volume si differenziano lungo quattro assi principali: (1) qualità dei campioni, (2) diversità/copertura della distribuzione, (3) velocità di campionamento, (4) possibilità di calcolare la likelihood.

Tabella comparativa dei framework generativi

Modello	Qualità	Copertura	Velocità	Likelihood
VAE	Media	Alta	$O(1)$	ELBO
Flow (RealNVP/Glow)	Alta	Alta	$O(1)$	Esatta
GAN	Alta	Bassa	$O(1)$	No
WGAN-GP	Alta	Media	$O(1)$	No
DDPM	Molto alta	Molto alta	$O(T)$	ELBO
DDIM	Alta	Alta	$O(\sqrt{T})$	No
ODE/CNF	Alta	Alta	$O(\text{NFE})$	Esatta
OT-CFM	Molto alta	Alta	$O(1\text{–}5)$	Appross.
EBM	Variabile	Alta	$O(T_{\text{MCMC}})$	No (NCE)

Trade-off qualità-velocità. I modelli di diffusione (DDPM) producono campioni di qualità stato dell'arte (FID $\approx$ 1–3 su ImageNet 256×256) ma richiedono 1000 forward pass del denoiser. I flow matching (OT-CFM) raggiungono qualità comparabile con 5–50 step. I modelli one-step come le GAN sono i più veloci ma tendono al mode collapse.

Copertura vs. qualità: il trade-off precision/recall. La *precision* misura la frazione di campioni generati che cadono nella distribuzione reale; il *recall* misura la frazione della distribuzione reale coperta dai campioni generati. GAN hanno alta precision ma basso recall (mode collapse). Diffusion models hanno alta precision e alto recall. VAE hanno recall alto (prior gaussiano = distribuzione uniforme sullo spazio latente) ma precision moderata.

Stabilità di training.

- **VAE/Flows/Diffusion**: training stabile per MLE o ELBO; la loss è unimodale (per VAE gaussiano) o monotona (ELBO diffusion).
- **GAN**: training a minimax con instabilità (oscillazioni, mode collapse, gradient vanishing). WGAN-GP è più stabile ma ancora non garantisce convergenza.
- **EBM**: training con MCMC richiede catene lunghe per stime accurate; CD introduce bias sistematico.

Condizionamento. Tutti i framework sono estensibili al caso condizionale $p(\mathbf{x}|\mathbf{c})$ (generazione condizionata da un testo, una classe, o un'immagine guida). I modelli di diffusione con Classifier-Free Guidance (CFG, Sezione 20.5) sono stati i più adottati per generazione testo-immagine (Stable Diffusion, DALL-E 3, Imagen 3).

Model	Exact $\log p$	Fast sampling	Training signal	Mode coverage
VAE	No (ELBO)	Yes (1 pass)	ELBO (lower bound)	Tends to average
GAN	No	Yes (1 pass)	Adversarial	Mode collapse risk
NF	Yes	Yes (1 pass)	MLE exact	Good, limited by arch.
Diffusion	No (ELBO)	No (T steps)	Denoising MSE	Excellent
Flow Match.	No (ELBO)	No (T ODE)	Regression	Excellent

Computational cost comparison.

Model	Train cost	Sample cost	Likelihood	Memory
VAE	$O(d)$ per step	$O(1)$ fwd pass	Lower bound	Low
GAN	$2\times$ fwd+bwd	$O(1)$ fwd pass	None	Low
Flow	$O(d)$ fwd+bwd	$O(1)$ fwd pass	Exact	Low
DDPM	$O(d)$ per step	$O(T \cdot d)$	Lower bound	Low train, High sample
CFM	$O(d)$ per step	$O(K \cdot d)$ $K \ll T$	Lower bound	Low
EBM	MCMC per step	MCMC	Unnormalised	Low
GP	$O(n^3)$ fit	$O(n^2)$ pred	Exact	$O(n^2)$

Sample quality vs. diversity tradeoff.

- *GANs*: highest FID (sharpest samples) but low diversity (mode collapse risk); no likelihood.
- *Diffusion*: near-GAN quality, high diversity, exact DDIM inversion; slow sampling mitigated by distillation (Consistency Models, ∼1–4 steps).
- *VAEs*: fast and stable; tend to produce blurry samples due to Gaussian decoder assumption.
- *Flows*: exact likelihood and invertibility; lower FID than VAE/GAN on structured data; memory-intensive for high-dimensional inputs.

Current landscape (2024–2025). Diffusion models dominate image and audio generation (Stable Diffusion, DALL-E 3, Sora). Flow matching is replacing DDPM as the training objective (Stable Diffusion 3, Flux). VAEs are used as encoders in *latent* diffusion models (compress images from $512 \times 512 \times 3$ to $64 \times 64 \times 4$ before diffusion). GANs remain competitive for video synthesis (StyleGAN-V) and real-time generation.

24.2 Theoretical Connections

Unificazione tramite score function. Song et al. (2021) mostrano che VAE, diffusion models e flow matching sono tutti casi speciali di un framework unificato basato sull'equazione di continuità e sullo score $\mathbf{s}_t(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$:

$$\begin{aligned} \text{Diffusion (SDE):} &\quad d\mathbf{x} = \mathbf{f}\, dt + g\, d\mathbf{w} \\ \text{Flow (ODE):} &\quad d\mathbf{x} = [\mathbf{f} - \tfrac{1}{2} g^2 \mathbf{s}_t]\, dt \\ \text{Score matching:} &\quad \mathbf{s}_\theta \approx \mathbf{s}_t \\ \text{Diffusion ELBO:} &\quad -\log p \le -\mathcal{L}_{\text{ELBO}} \propto \mathbb{E}\|\mathbf{s}_\theta - \mathbf{s}_t\|_w^2 \end{aligned}$$

Connessione VAE ↔ diffusion. Un VAE con encoder e decoder gaussiani può essere visto come un modello di diffusione a un solo passo: il processo forward è $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}^2_\phi)$ (un passo di rumore adattivo), il processo reverse è $p_\theta(\mathbf{x}|\mathbf{z})$ (un passo di denoising). La DDPM con $T \to \infty$ step è il limite continuo del VAE gerarchio profondo (HVAE).

Connessione GAN ↔ flow. Un generatore GAN $G_\theta(\mathbf{z})$ è un normalizing flow implicito: definisce una trasformazione invertibile (in linea di principio) da $p(\mathbf{z})$ a p_g. La differenza è che le GAN non richiedono l'invertibilità esplicita e non calcolano il Jacobiano; il discriminatore fornisce il segnale di addestramento invece della log-likelihood.

Connessione score matching ↔ EBM. The score of an EBM is $\mathbf{s}_\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} E_\theta(\mathbf{x})$ — linearly

proportional to the energy gradient. Training $\mathbf{s}_\theta$ with DSM is equivalent to estimare $\nabla_{\mathbf{x}} E_\theta$ senza calcolare $Z(\theta)$.

Gerarchia di trattabilità.

1. **Esattamente trattabili**: flows (normalizing flows con Jacobiano calcolabile), GPR su dataset piccoli.
2. **ELBO trattabile**: VAE, DDPM, DGP (con inducing points).
3. **Solo campioni, no likelihood**: GAN, WGAN, modelli di diffusione con DDIM, OT-CFM (approssimato).
4. **Intrattatabile, MCMC necessario**: EBM, exact BNN posterior.

Framework teorico unificato: SB (Schrödinger Bridge). Il problema di Schrödinger Bridge (SB) trova il processo stocastico di minima entropia relativa che porta p_0 a p_1. Il CFM è il caso deterministico ($\sigma \to 0$), DDPM è un caso stocastico con prior fisso, e l'SB generalizza entrambi. Modelli come DiffBridge e I^{2}SB operano direttamente nello spazio SB per generazione image-to-image.

All generative models ultimately optimise some divergence or distance between p_θ and p_{data}:

Model	**Implicit divergence minimised**
VAE	$D_{\text{KL}}\big(q_\phi(\mathbf{z}\,\vert\,\mathbf{x}) \,\Vert\, p_\theta(\mathbf{z}\,\vert\,\mathbf{x})\big)$ at each step
GAN (vanilla)	Jensen-Shannon divergence $\text{JSD}(p_{\text{data}} \Vert p_\theta)$
WGAN	Wasserstein-1 distance $W_1(p_{\text{data}}, p_\theta)$
NF (MLE)	$D_{\text{KL}}(p_{\text{data}} \,\Vert\, p_\theta)$ (forward KL)
Diffusion	Score MSE $\sim D_{\text{KL}}(p_{\text{data}} \,\Vert\, p_\theta)$ (in the limit)

Mode-seeking vs. mass-covering. $D_{\text{KL}}(p_\theta \,\Vert\, p_{\text{data}})$ (reverse KL, VAEs) is mode-seeking: p_θ concentrates where $p_{\text{data}} > 0$ to avoid $\log(p_\theta / p_{\text{data}}) \to +\infty$. $D_{\text{KL}}(p_{\text{data}} \,\Vert\, p_\theta)$ (forward KL, MLE) is mass-covering: p_θ must assign density wherever $p_{\text{data}} > 0$.

Diffusion as score matching. The DDPM training objective $\|\epsilon_\theta(\mathbf{x}_t, t) - \epsilon\|^2$ is equivalent to denoising score matching at noise level σ_t, as shown in §20.6. This connects DDPM to EBMs (§22.1–22.2): the noise predictor ϵ_θ implicitly learns the gradient of the log-density (score) of the noisy data distribution.

VAE as a special case of diffusion. A VAE with a single latent $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and decoder $p_\theta(\mathbf{x}|\mathbf{z})$ can be viewed as a 1-step diffusion model: encode to noise, decode from noise. DDPM generalises this to T steps, distributing the encoding/decoding across a Markov chain.

Normalising flows vs. diffusion.

Model	Exact likelihood	FID	Sampling speed	Inversion
Normalising flow	Yes	~ 30	Fast	Yes
VAE	Lower bound	~ 20	Fast	Approx
GAN	No	~ 2–8	Fast	No
Diffusion	Lower bound	~ 2–5	Slow ($\sim$steps)	DDIM only

InfoNCE and variational MI. The InfoNCE lower bound on $I(\mathbf{x}; \mathbf{c})$ (§22.4) can be derived as a special case of the ELBO for a latent variable model where the posterior is approximated by a K-class softmax. This connects self-supervised learning (CLIP, SimCLR) to the variational inference framework of Part 16.

Energy-based unification. All generative models can be cast as EBMs: a flow defines $E(\mathbf{x}) = -\log p_Z(f^{-1}(\mathbf{x})) - \log|\det J_{f^{-1}}|$; a VAE defines $E(\mathbf{x}) = -\mathcal{L}_{\text{ELBO}}(\mathbf{x})$; DDPM defines $E(\mathbf{x}) = \|\mathbf{x} - \mu_\theta(\mathbf{x}_1, 1)\|^2/2\sigma_1^2$ (approximately). The EBM framing clarifies that all these models are learning an implicit or explicit energy landscape over the data space.

VAE–Flow equivalence. A VAE with a sufficiently expressive flow-based decoder and encoder (normalising flows in both q_ϕ and p_θ) can represent any pair of marginal $p(\mathbf{x})$ and posterior $p(\mathbf{z}|\mathbf{x})$. In the limit of infinite capacity, optimising the ELBO with flow-based components converges to the same solution as maximum likelihood with a normalising flow. This shows that the VAE–NF distinction is architectural, not fundamental: both are maximum likelihood methods with different parameterisations of the density.

GAN–EBM duality. The GAN discriminator $D(\mathbf{x})$ defines an unnormalised density $p_D(\mathbf{x}) \propto \exp(D(\mathbf{x}))$ at optimality ($D^*(\mathbf{x}) = \log \frac{p_{\text{data}}(\mathbf{x})}{p_G(\mathbf{x})} + c$). Training the discriminator is equivalent to fitting a ratio estimator p_{data}/p_G. From this perspective, the GAN generator performs one step of contrastive divergence on the energy $-D(\mathbf{x})$: moving mass from regions of low D to high D, i.e. from the model's current distribution toward the data distribution.

Diffusion as hierarchical VAE. DDPM can be written as a VAE with T latent variables $\{\mathbf{x}_t\}_{t=1}^T$. The encoder is the fixed Gaussian forward process $q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_t q(\mathbf{x}_t|\mathbf{x}_{t-1})$, and the decoder is the learned reverse $p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_t p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. The ELBO becomes:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_q[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)] - \sum_{t=2}^{T} \mathbb{E}_q[D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)\|p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))] - D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\|p(\mathbf{x}_T))$$

This framing explains why diffusion models succeed: the fixed encoder (forward process) provides a rich, well-calibrated posterior for all T latent levels, avoiding the posterior collapse and difficult amortisation problems of standard VAEs.

Score matching and flow matching unification. Flow matching (§21.2) learns a velocity field $v_\theta(\mathbf{x}, t)$ interpolating between noise and data. The probability flow ODE of diffusion (§20.4) also defines a velocity field: $v(\mathbf{x}, t) = f(\mathbf{x}, t) - \frac{1}{2}g(t)^2\nabla_{\mathbf{x}}\log p_t(\mathbf{x})$. At the level of the ODE, score-based diffusion and flow matching are identical — both learn to push a simple initial distribution to the data distribution along a deterministic trajectory. The difference is in the training objective: score matching trains with DSM noise, flow matching with simulation-free regression on interpolant paths. Flow matching is more general (accommodating OT paths) and often more stable.

24.3 Summary of Volume II

The common thread. This volume has developed, step by step, the mathematics of deep generative models starting from probabilistic foundations. The unifying thread is the ELBO: every model treated is, in a precise sense, a specialisation of the variational objective $\log p(\mathbf{x}) \geq \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z})] - \mathbb{E}_q[\log q(\mathbf{z})]$.

Part 13 — Probabilistic foundations.

- Probability spaces and measures: the rigorous foundation on which all inference rests.
- Expectation and inequalities (Jensen, Markov, Chebyshev, Hoeffding): tools for proving bounds and controlling estimator variance.
- KL Divergence: the fundamental measure between distributions; always non-negative, asymmetric, non-metric.
- Bayes' theorem: maps from likelihood to posterior; the evidence $p(\mathcal{D})$ is the central intractability node.

Parts 14–16 — Classical variational inference.

- The exponential family unifies Gaussian, Bernoulli, Poisson, categorical; the conjugate prior guarantees closed-form updates.
- Directed graphical models (DAGs) encode conditional independencies; d-separation is the algorithmic criterion.
- The ELBO+KL decomposition (§15.1) is the foundation; EM maximises the ELBO by alternating block updates (E-step = optimise q, M-step = optimise θ).
- CAVI: coordinate ascent in the mean-field family; convergent, but only to local optima.

Parts 17–19 — Deep models: VAE and GAN.

- The VAE (Kingma & Welling, 2014) resolves CAVI intractability by amortising inference with a neural encoder and using the reparametrisation trick for gradients.
- Normalising flows (§18) provide exact likelihood via the change-of-variables formula; triangular Jacobians make computation scalable.
- GANs (§19) replace the likelihood with a minimax game; WGAN uses the Wasserstein distance for more stable gradients.

Parts 20–22 — Diffusion, flows, EBMs.

- Diffusion models (Ho et al., 2020) define a forward process of gradual corruption and learn the reverse; the objective reduces to predicting the noise $\boldsymbol{\epsilon}$.
- The Probability Flow ODE transforms stochastic diffusion into a deterministic flow, enabling fast sampling (DDIM) and exact inversion.
- Conditional Flow Matching (Lipman et al., 2022) avoids ODE simulation during training; OT-CFM produces straight trajectories and NFE ≈ 5.
- EBMs define densities via energy; score matching and DSM avoid the partition function.

Parts 23–24 — Gaussian processes and synthesis.

- GPs provide a non-parametric functional prior; GP regression is exact Bayesian inference.
- Deep GPs compose GP layers for increased flexibility; inference uses variational inducing points.
- All frameworks unify under the score-based paradigm: score function estimation is the fundamental operation.

Connections with Volume I (architectures). Volume I built neural networks as parametric functions f_θ; this volume uses them as:

- **Encoder/decoder in the VAE**: $q_\phi(\mathbf{z}|\mathbf{x})$, $p_\theta(\mathbf{x}|\mathbf{z})$.
- **Coupling layers in flows**: $s_\theta(z_{1:m})$, $t_\theta(z_{1:m})$ in RealNVP.
- **Denoiser in diffusion models**: $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ or U-Net with attention.
- **Velocity field in flow matching**: $\mathbf{v}_\theta(\mathbf{z}_t, t)$.
- **Energy function in EBMs**: $E_\theta(\mathbf{x})$.

The architecture of choice varies: MLP for tabular data, CNN/ResNet for images, Transformer for sequences and high-dimensional data. Volume III will develop sequence-based architectures (RNN, LSTM, Transformer) that underpin generative models for language and serial data.

This volume has developed the complete mathematical theory of probabilistic and generative models:

- **Part 13:** Measure-theoretic probability, KL divergence, Bayes' theorem.
- **Part 14:** Exponential family, graphical models, latent variable models.
- **Part 15:** EM algorithm — derivation, E/M-step, monotone convergence proof.
- **Part 16:** Variational inference — ELBO, mean-field, CAVI.
- **Part 17:** VAE — generative model, inference network, reparameterisation trick, full training step.
- **Part 18:** Normalising flows — change-of-variables formula, autoregressive flows, Jacobian structure.
- **Part 19:** GANs — minimax objective, optimal discriminator, JSD interpretation, WGAN with Wasserstein distance and gradient penalty.
- **Part 20:** Diffusion models — forward process, closed-form marginals, ELBO decomposition, noise prediction objective, probability flow ODE, classifier-free guidance.
- **Part 21:** Flow matching — continuity equation, CFM objective, OT-CFM with straight-line trajectories.
- **Part 22:** Energy-based models, score matching, denoising score matching.
- **Part 23:** Gaussian processes and deep GPs, variational inference for deep GPs.
- **Part 24:** Unified comparison of generative frameworks.

Volume III will cover graph neural networks, geometric deep learning, and multimodal architectures.

Mathematics of Neural Networks — Volume II of V
Silvano Salvador

Recap by part.

Part 13 (Probability & Information): Probability spaces, expectation, KL divergence, conditional probability, Shannon entropy/mutual information, Fisher information and Cramér-Rao bound. These tools underpin all probabilistic modelling.

Part 14 (Graphical Models): Exponential family, directed Bayesian networks, undirected MRFs, hierarchical Bayesian models. The language for describing structured probabilistic dependencies.

Part 15 (EM Algorithm): ELBO as a free energy, E-step and M-step, EM for GMMs, generalised EM (VEM, MCEM). The prototypical algorithm for latent variable ML.

Part 16 (Variational Inference): ELBO derivation and three forms, mean-field approximation, CAVI, black-box VI, amortised inference. The approximation framework connecting classical Bayesian inference to neural networks.

Part 17 (VAE): Generative model and decoder families, inference network and ELBO, reparameterisation trick, complete training step, β-VAE, posterior collapse. The canonical deep generative model combining neural networks and variational inference.

Part 18 (Normalising Flows): Change-of-variables formula, autoregressive flows (MAF/IAF), coupling layers and RealNVP, Glow. Models with exact likelihood via bijective transformations.

Part 19 (GANs): Minimax objective and JSD interpretation, WGAN and Wasserstein distance, training dynamics and mode collapse, spectral normalisation. Implicit generative models via adversarial training.

Part 20 (Diffusion Models): Forward process, reverse process ELBO, DDPM training objective, probability flow ODE, classifier-free guidance, score-based models, DDIM. The dominant paradigm for high-fidelity image/audio/video synthesis.

Part 21 (Flow Matching): Continuous normalising flows, CFM objective, OT-CFM. The efficient successor to DDPM for straight-line generative trajectories.

Part 22 (EBMs): Energy functions, score matching (explicit, sliced, denoising), contrastive divergence, noise contrastive estimation. The unifying perspective connecting all generative models as energy landscapes.

Part 23 (GPs): GP definition, marginalisation, regression, deep GPs, GP regression derivation, kernel functions, NTK. Bayesian nonparametric regression and its connection to infinite neural networks.

Part 24 (Synthesis): Comparison of generative frameworks, theoretical connections (KL/JSD/Wasserstein, ELBO/score, VAE/diffusion/flows), landscape of current models.

Volume III continues with *Representation Learning*: self-supervised methods (contrastive, masked modelling, BYOL), transfer learning, foundation models, and the formal study of generalisation (PAC learning, VC dimension, Rademacher complexity). The probabilistic

foundations of Volume II — particularly mutual information, KL divergence, and variational inference — are prerequisites for understanding the information-theoretic aspects of representation learning.

Markov Chain Monte Carlo and Langevin Dynamics

This appendix collects the MCMC methods referenced throughout the volume — Gibbs sampling, Metropolis–Hastings, and Langevin dynamics — and provides the convergence guarantees used in the EBM and score-based sections.

A.1 Markov Chains and Stationary Distributions

A *Markov chain* on state space $\mathcal{X}$ is a sequence $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \dots$ with transition kernel $T(\mathbf{x}' \mid \mathbf{x})$ satisfying the Markov property: $p(\mathbf{x}^{(t+1)} \mid \mathbf{x}^{(0)}, \dots, \mathbf{x}^{(t)}) = T(\mathbf{x}^{(t+1)} \mid \mathbf{x}^{(t)})$.

Stationary distribution

A distribution π is *stationary* for T if $\pi(\mathbf{x}') = \int T(\mathbf{x}' \mid \mathbf{x})\, \pi(\mathbf{x})\, d\mathbf{x}$ for all $\mathbf{x}'$. Sufficient condition: *detailed balance* $\pi(\mathbf{x}) T(\mathbf{x}' \mid \mathbf{x}) = \pi(\mathbf{x}') T(\mathbf{x} \mid \mathbf{x}')$ for all $\mathbf{x}, \mathbf{x}'$.

A chain is *ergodic* (irreducible + aperiodic) iff it has a unique stationary distribution π and $\| p^{(t)} - \pi \|_{\mathrm{TV}} \to 0$ as $t \to \infty$ for any initialisation $p^{(0)}$.

A.2 Metropolis–Hastings Algorithm

Given a target $\pi(\mathbf{x}) \propto \tilde{\pi}(\mathbf{x})$ (unnormalised) and a proposal $q(\mathbf{x}' \mid \mathbf{x})$:

1. At state $\mathbf{x}^{(t)}$, propose $\mathbf{x}' \sim q(\cdot \mid \mathbf{x}^{(t)})$.
2. Compute acceptance ratio:

$$\alpha = \min\left(1,\ \frac{\tilde{\pi}(\mathbf{x}')\, q(\mathbf{x}^{(t)} \mid \mathbf{x}')}{\tilde{\pi}(\mathbf{x}^{(t)})\, q(\mathbf{x}' \mid \mathbf{x}^{(t)})}\right)$$

3. Accept: $\mathbf{x}^{(t+1)} = \mathbf{x}'$ with probability α; else $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)}$.

The resulting chain satisfies detailed balance w.r.t. π. For a symmetric proposal $q(\mathbf{x}' \mid \mathbf{x}) = q(\mathbf{x} \mid \mathbf{x}')$ (Metropolis), the ratio simplifies to $\tilde{\pi}(\mathbf{x}')/\tilde{\pi}(\mathbf{x}^{(t)})$.

Optimal acceptance rate. For a d-dimensional Gaussian target with isotropic Gaussian proposal $q(\mathbf{x}' \mid \mathbf{x}) = \mathcal{N}(\mathbf{x}', \mathbf{x}, \sigma^2 \mathbf{I})$, the optimal $\sigma^2 = 2.38^2/d$ achieves $\approx 23.4\%$ acceptance rate and maximises the effective sample size per unit computation (Roberts & Rosenthal, 1998).

A.3 Gibbs Sampling

Gibbs sampling is a special case of MH with acceptance rate $\equiv 1$. At each step, sample one variable from its full conditional:

$$x_i^{(t+1)} \sim p(x_i \mid \mathbf{x}_{-i}^{(t)})$$

For graphical models with local Markov structure, the full conditional $p(x_i \mid \mathbf{x}_{-i})$ depends only on the Markov blanket of x_i, making sampling efficient. For RBMs (§22.3), the conditional factorises over all units simultaneously — enabling *block* Gibbs sampling (all hidden units in parallel given visible, and vice versa).

Systematic vs. random scan. Systematic scan updates variables in a fixed order; random scan selects i uniformly at random. Both have π as stationary distribution; random scan is easier to analyse theoretically.

A.4 Unadjusted Langevin Algorithm (ULA)

The *Langevin SDE* with target $\pi(\mathbf{x}) \propto \exp(-U(\mathbf{x}))$ is:

$$d\mathbf{x} = -\nabla U(\mathbf{x})\, dt + \sqrt{2}\, d\mathbf{w}$$

Its stationary distribution is π (by the Fokker–Planck equation). Euler–Maruyama discretisation gives the *Unadjusted Langevin Algorithm*:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta\, \nabla U(\mathbf{x}^{(t)}) + \sqrt{2\eta}\, \mathbf{z}^{(t)}, \quad \mathbf{z}^{(t)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

ULA is *biased*: the stationary distribution of the discrete chain is $\pi_\eta \neq \pi$ due to discretisation error. The bias is $O(\eta)$ in total variation: $\|\pi_\eta - \pi\|_{\mathrm{TV}} = O(\eta)$ for log-smooth targets.

Metropolis-Adjusted Langevin Algorithm (MALA). Adding an MH correction step removes the bias:

$$\mathbf{x}' = \mathbf{x}^{(t)} - \eta \nabla U(\mathbf{x}^{(t)}) + \sqrt{2\eta}\, \mathbf{z}, \quad \alpha = \min\left(1, \frac{\pi(\mathbf{x}')\, q(\mathbf{x}^{(t)}|\mathbf{x}')}{\pi(\mathbf{x}^{(t)})\, q(\mathbf{x}'|\mathbf{x}^{(t)})}\right)$$

MALA has exact stationary distribution π and mixes faster than MH by a factor of $O(d^{1/3})$ in dimension d.

A.5 Convergence under Log-Sobolev Inequality

Log-Sobolev inequality (LSI)

Distribution π satisfies LSI with constant $\rho > 0$ if for all smooth $f > 0$:

$$\mathrm{Ent}_\pi(f) \triangleq \mathbb{E}_\pi[f \log f] - \mathbb{E}_\pi[f] \log \mathbb{E}_\pi[f] \;\leq\; \frac{1}{\rho} \mathbb{E}_\pi\left[\frac{\|\nabla f\|^2}{f}\right]$$

Under LSI(ρ), the Langevin SDE satisfies exponential convergence in KL:

$$D_{\mathrm{KL}}(p_t \,\|\, \pi) \leq e^{-2\rho t}\, D_{\mathrm{KL}}(p_0 \,\|\, \pi)$$

Proof: differentiate $D_{\mathrm{KL}}(p_t \,\|\, \pi)$ using the Fokker–Planck equation, apply LSI, integrate. Mixing time to ϵ-error: $t_{\mathrm{mix}} = O(\rho^{-1} \log D_{\mathrm{KL}}(p_0 \,\|\, \pi)\, / \epsilon)$.

Examples. $\mathcal{N}(\square, \Sigma)$ satisfies LSI with $\rho = \lambda_{\min}(\Sigma^{-1})$. Uniform on a convex body satisfies LSI (Kannan–Lovász–Simonovits conjecture, proved for isotropic log-concave measures). Multimodal distributions (e.g. GMM with well-separated modes) have $\rho \approx e^{-\Delta E/2}$ where ΔE is the energy barrier — exponentially slow mixing.

A.6 Hamiltonian Monte Carlo (HMC)

HMC introduces auxiliary momentum variables $\mathbf{p} \sim \mathcal{N}(\mathbf{0}, \mathbf{M})$ and simulates Hamiltonian dynamics:

$$H(\mathbf{x}, \mathbf{p}) = U(\mathbf{x}) + \frac{1}{2} \mathbf{p}^\top \mathbf{M}^{-1} \mathbf{p}$$

The leapfrog integrator (time-reversible, volume-preserving) is used for numerical integration, followed by an MH acceptance step. HMC generates proposals far from the current state with high acceptance probability. Mixing time scales as $O(d^{1/4})$ vs. $O(d)$ for random-walk MH and $O(d^{1/3})$ for MALA — a significant advantage in high dimensions.

Relevance to deep generative models. ULA is the continuous-time limit of the denoising diffusion reverse process (§20.1–20.2): the reverse SDE is a Langevin equation targeting p_0 (data distribution) with drift computed from the learned score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. Convergence guarantees for the reverse SDE (e.g. De Bortoli et al., 2022) follow directly from the Langevin convergence theory above, under smoothness assumptions on p_0.

Elements of Measure-Theoretic Probability

This appendix provides the measure-theoretic foundations underlying the probabilistic concepts used throughout the volume, particularly in Part 13 and Part 16.

B.1 σ-Algebras and Measurable Spaces

Measurable space

A *measurable space* $(\Omega, \mathcal{F})$ consists of a sample space Ω and a σ-algebra $\mathcal{F} \subseteq 2^\Omega$, i.e. a collection of subsets of Ω closed under complements and countable unions. Elements of $\mathcal{F}$ are called *measurable sets* or *events*.

Standard σ-algebras.

- **Borel σ-algebra** $\mathcal{B}(\mathbb{R}^d)$: the smallest σ-algebra containing all open subsets of $\mathbb{R}^d$. Contains all open, closed, countable, and uncountable "nice" subsets of $\mathbb{R}^d$.
- **Product σ-algebra** $\mathcal{F}_1 \otimes \mathcal{F}_2$: generated by $\{A_1 \times A_2 : A_i \in \mathcal{F}_i\}$. Used for joint distributions.
- **Lebesgue σ-algebra**: completion of $\mathcal{B}(\mathbb{R}^d)$ under the Lebesgue measure — contains sets of Lebesgue measure zero.

B.2 Probability Measures and Random Variables

Probability measure

A *probability measure* on $(\Omega, \mathcal{F})$ is a function $P : \mathcal{F} \to [0,1]$ satisfying (i) $P(\Omega) = 1$, (ii) $P(\emptyset) = 0$, (iii) σ-additivity: $P(\bigcup_{n=1}^{\infty} A_n) = \sum_{n=1}^{\infty} P(A_n)$ for pairwise disjoint $A_n \in \mathcal{F}$.

A *random variable* $X : (\Omega, \mathcal{F}) \to (\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$ is a measurable function: $X^{-1}(B) \in \mathcal{F}$ for all $B \in \mathcal{B}(\mathbb{R}^d)$. The *law* (pushforward measure) of X is $P_X(B) = P(X^{-1}(B))$.

Density vs. distribution. P_X has a *probability density function* (pdf) $p_X : \mathbb{R}^d \to [0, \infty)$ w.r.t. Lebesgue measure λ iff it is absolutely continuous: $P_X \ll \lambda$, i.e. $\lambda(B) = 0 \Rightarrow P_X(B) = 0$. By the Radon–Nikodym theorem, $P_X(B) = \int_B p_X(\mathbf{x})\, d\mathbf{x}$ in this case.

B.3 Expectation and Integration

The *expectation* of a measurable function $f : \Omega \to \mathbb{R}$ is the Lebesgue integral:

$$\mathbb{E}[f(X)] = \int_\Omega f(X(\omega))\, P(d\omega) = \int_{\mathbb{R}^d} f(\mathbf{x})\, p_X(\mathbf{x})\, d\mathbf{x}$$

(the last equality holds when $P_X \ll \lambda$).

Key theorems.

- **Monotone convergence (MCT):** if $0 \le f_n \uparrow f$ pointwise, $\mathbb{E}[f_n] \uparrow \mathbb{E}[f]$. Used to exchange limits and integrals in EM proofs.
- **Dominated convergence (DCT):** if $|f_n| \le g$ with $\mathbb{E}[g] < \infty$ and $f_n \to f$ a.e., then $\mathbb{E}[f_n] \to \mathbb{E}[f]$.

- **Fubini–Tonelli:** for product measures, $\int f\, d(P \otimes Q) = \int\int f\, dP\, dQ$ when $f \geq 0$ or $\mathbb{E}[|f|] < \infty$.

B.4 Radon–Nikodym Theorem and KL Divergence

Radon–Nikodym theorem

If $\mu \ll \nu$ (i.e. μ is absolutely continuous w.r.t. ν), there exists a unique (a.e.) non-negative measurable function $\frac{d\mu}{d\nu}$ (the *Radon–Nikodym derivative* or *density ratio*) such that $\mu(A) = \int_A \frac{d\mu}{d\nu}\, d\nu$ for all measurable A.

The KL divergence (§13.3) is defined for any $p \ll q$ via the Radon–Nikodym derivative:

$$D_{\mathrm{KL}}(p \,\|\, q) = \int \frac{dp}{dq} \log \frac{dp}{dq}\, dq = \mathbb{E}_p\left[\log \frac{dp}{dq}\right]$$

This definition is coordinate-free: it does not require p, q to have densities w.r.t. Lebesgue measure, only that $p \ll q$. For continuous distributions on $\mathbb{R}^d$ with densities $p(\mathbf{x}), q(\mathbf{x})$, this reduces to the familiar integral.

Non-negativity (Gibbs' inequality). $D_{\mathrm{KL}}(p \,\|\, q) \geq 0$ with equality iff $p = q$ μ-a.e. Proof: apply Jensen's inequality to $-\log$:

$$-D_{\mathrm{KL}}(p \,\|\, q) = \mathbb{E}_p\left[\log \frac{q}{p}\right] \leq \log \mathbb{E}_p\left[\frac{q}{p}\right] = \log \int q\, d\mu = \log 1 = 0$$

B.5 Conditional Expectation

For σ-algebras $\mathcal{G} \subseteq \mathcal{F}$, the *conditional expectation* $\mathbb{E}[X|\mathcal{G}]$ is the unique $\mathcal{G}$-measurable random variable satisfying $\int_A \mathbb{E}[X|\mathcal{G}]\, dP = \int_A X\, dP$ for all $A \in \mathcal{G}$. Existence follows from the Radon–Nikodym theorem.

Tower property. For $\mathcal{H} \subseteq \mathcal{G} \subseteq \mathcal{F}$: $\mathbb{E}[\mathbb{E}[X|\mathcal{G}]|\mathcal{H}] = \mathbb{E}[X|\mathcal{H}]$. This underlies the EM ELBO derivation: $\mathbb{E}_{\mathbf{z}}[\mathbb{E}_{\mathbf{x}|\mathbf{z}}[\cdot]] = \mathbb{E}_{\mathbf{x},\mathbf{z}}[\cdot]$.

Disintegration theorem. For a joint distribution $P_{X,Z}$ with marginal P_Z, the conditional distribution $P_{X|Z=\mathbf{z}}$ exists as a probability kernel (regular conditional probability) under mild measurability conditions. This justifies writing $p(\mathbf{x}|\mathbf{z})$ as a well-defined density for (Lebesgue-)a.e. $\mathbf{z}$.

Stochastic Optimisation and Gradient Estimation

This appendix collects gradient estimation techniques used in variational inference (§16.4) and the VAE reparameterisation trick (§17.3), and surveys the main stochastic optimisation algorithms used in training deep generative models.

C.1 Score-Function Estimator (REINFORCE)

For $\mathbb{E}_{p_\phi(\mathbf{z})}[f(\mathbf{z})]$ with ϕ-dependent distribution, the *score-function estimator* (Williams, 1992) is:

$$\nabla_\phi \, \mathbb{E}_{p_\phi}[f(\mathbf{z})] = \mathbb{E}_{p_\phi}\left[f(\mathbf{z}) \, \nabla_\phi \log p_\phi(\mathbf{z})\right]$$

Proof: $\nabla_\phi \int f(\mathbf{z}) p_\phi(\mathbf{z}) d\mathbf{z} = \int f(\mathbf{z}) \nabla_\phi p_\phi(\mathbf{z}) d\mathbf{z} = \int f(\mathbf{z}) \, p_\phi(\mathbf{z}) \, \nabla_\phi \log p_\phi(\mathbf{z}) \, d\mathbf{z}$.

Variance reduction. The score function estimator has high variance: the gradient signal is multiplied by $f(\mathbf{z})$, which may be large and noisy. Control variates reduce variance: $f(\mathbf{z}) - b$ for any baseline b with $\mathbb{E}[\nabla_\phi \log p_\phi(\mathbf{z}) \, b] = 0$ (which holds for any constant b). Optimal baseline: $b^* = \mathbb{E}[f(\mathbf{z}) \|\nabla_\phi \log p_\phi\|^2] / \mathbb{E}[\|\nabla_\phi \log p_\phi\|^2]$.

C.2 Reparameterisation Gradient

When $\mathbf{z} \sim p_\phi(\mathbf{z})$ can be written as $\mathbf{z} = g_\phi(\square)$ with $\square \sim p(\square)$ (ϕ-independent):

$$\nabla_\phi \, \mathbb{E}_{p_\phi}[f(\mathbf{z})] = \mathbb{E}_{p(\square)}\left[\nabla_\phi f(g_\phi(\square))\right] = \mathbb{E}_{p(\square)}\left[\frac{\partial f}{\partial \mathbf{z}} \frac{\partial g_\phi}{\partial \phi}\right]$$

This "pathwise" estimator has orders-of-magnitude lower variance than the score function estimator because the gradient flows directly through the sample path $\mathbf{z} = g_\phi(\square)$.

Applicability. Valid when p_ϕ is reparameterisable: Gaussian ($\mathbf{z} = \square + \square \odot \square$), Laplace, logistic, uniform, Gamma (shape > 1). *Not* directly applicable to discrete distributions; workarounds: Gumbel–Softmax (Jang et al., 2017), straight-through estimator.

Gumbel–Softmax trick. For a categorical variable with logits $\square$, the reparameterised sample is:

$$\mathbf{y} = \text{softmax}((\log \square + \mathbf{g})/\tau), \quad g_k \sim \text{Gumbel}(0,1) = -\log(-\log u_k), \; u_k \sim \mathcal{U}[0,1]$$

As $\tau \to 0$, $\mathbf{y}$ approaches a one-hot vector; for finite τ, the gradient $\partial \mathbf{y} / \partial \square$ is well-defined. Straight-through: use the hard one-hot in the forward pass but the soft Gumbel–Softmax gradient in the backward pass.

C.3 Adam and Adaptive Gradient Methods

Training deep generative models uses *adaptive* gradient methods that scale the learning rate per-parameter.

Adam (Kingma & Ba, 2015)

Given gradient $\mathbf{g}_t = \nabla_\theta \mathcal{L}(\theta_{t-1})$ at step t:

$$\begin{aligned}
\mathbf{m}_t &= \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t \quad \text{(first moment estimate)} \\
\mathbf{v}_t &= \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2 \quad \text{(second moment estimate)} \\
\hat{\mathbf{m}}_t &= \mathbf{m}_t / (1-\beta_1^t), \quad \hat{\mathbf{v}}_t = \mathbf{v}_t / (1-\beta_2^t) \quad \text{(bias correction)} \\
\theta_t &= \theta_{t-1} - \eta \, \hat{\mathbf{m}}_t / (\sqrt{\hat{\mathbf{v}}_t} + \epsilon)
\end{aligned}$$

Defaults: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $\eta = 10^{-4}$ (generative models typically use $\eta \in [10^{-5}, 10^{-3}]$).

AdamW. Weight decay is correctly applied to the parameter, not to the gradient: $\theta_t \leftarrow (1 - \lambda)\theta_{t-1} - \eta\,\hat{\mathbf{m}}_t/(\sqrt{\hat{\mathbf{v}}_t} + \epsilon)$. Standard Adam conflates weight decay and L_2 regularisation; AdamW (Loshchilov & Hutter, 2019) decouples them, improving generalisation.

C.4 Learning Rate Schedules

Cosine annealing. Widely used for diffusion models and flow matching:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{\pi t}{T}\right)\right)$$

with T the total number of steps. In practice, $\eta_{\min} = 0$ and $\eta_{\max} = 2 \times 10^{-4}$ are typical.

Warmup. Linear warmup for the first W steps prevents gradient explosion at initialisation: $\eta_t = t/W \cdot \eta_{\max}$ for $t \leq W$. Used in all major diffusion model codebases (DDPM, EDM, Stable Diffusion).

Exponential Moving Average (EMA) of weights. A separate EMA copy of parameters $\bar{\theta}_t = \mu\bar{\theta}_{t-1} + (1 - \mu)\theta_t$ (with $\mu \in [0.999, 0.9999]$) is maintained and used exclusively for evaluation/sampling. EMA smooths out training noise and consistently improves sample quality (FID reduction of 10–30% in DDPM experiments).

C.5 Gradient Clipping

Large gradient norms destabilise training, particularly for recurrent models and diffusion models with many noise levels. Global gradient clipping:

$$\mathbf{g} \leftarrow \mathbf{g} \cdot \min\left(1,\ \frac{c}{\|\mathbf{g}\|_2}\right)$$

clips all parameter gradients to a maximum norm c. Typical values: $c = 1.0$ (DDPM, DiT) or $c = 0.5$ (text models). Clipping prevents individual large-loss samples from dominating parameter updates; combined with EMA, it is the primary stability mechanism in large-scale generative model training.

References

[1] M. S. Albergo and E. Vanden-Eijnden (2023). Building normalizing flows with stochastic interpolants. *ICLR 2023*.

[2] C. M. Bishop (2006). *Pattern Recognition and Machine Learning*. Springer.

[3] A. Brock, J. Donahue, and K. Simonyan (2019). Large scale GAN training for high fidelity natural image synthesis. *ICLR 2019*.

[4] Y. Burda, R. B. Grosse, and R. Salakhutdinov (2016). Importance weighted autoencoders. *ICLR 2016*.

[5] V. De Bortoli, J. Thornton, J. Heng, and A. Doucet (2022). Diffusion Schrödinger bridge with applications to score-based generative modeling. *NeurIPS 2022*.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

[7] P. Dhariwal and A. Q. Nichol (2021). Diffusion models beat GANs on image synthesis. *NeurIPS 2021*.

[8] L. Dinh, J. Sohl-Dickstein, and S. Bengio (2017). Density estimation using real-valued non-volume preserving (Real NVP) transformations. *ICLR 2017*.

[9] Y. Du and I. Mordatch (2019). Implicit generation and modeling with energy-based models. *NeurIPS 2019*.

[10] K. Fatras, Y. Zine, S. Majewski, R. Flamary, R. Gribonval, and N. Courty (2021). Minibatch optimal transport distances; analysis and applications. *arXiv:2101.01792*.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). Generative adversarial nets. *NeurIPS 2014*.

[12] M. Gutmann and A. Hyvärinen (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. *AISTATS 2010*.

[13] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter (2017). GANs trained by a two time-scale update rule converge to a local Nash equilibrium. *NeurIPS 2017*.

[14] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner (2017). β-VAE: Learning basic visual concepts with a constrained variational framework. *ICLR 2017*.

[15] G. E. Hinton (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

[16] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel (2019). Flow++: Improving flow-based generative models with variational dequantization and architecture design. *ICML 2019*.

[17] J. Ho, A. Jain, and P. Abbeel (2020). Denoising diffusion probabilistic models. *NeurIPS 2020*.

[18] J. Ho and T. Salimans (2022). Classifier-free diffusion guidance. *NeurIPS 2021 Workshop on DGMs*.

[19] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347.

[20] A. Hyvärinen (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–709.

[21] E. Jang, S. Gu, and B. Poole (2017). Categorical reparameterization with Gumbel-softmax. *ICLR 2017*.

[22] Y. Kim, S. Wiseman, A. C. Miller, D. Sontag, and A. M. Rush (2018). Semi-amortized variational autoencoders. *ICML 2018*.

[23] D. P. Kingma and J. Ba (2015). Adam: A method for stochastic optimization. *ICLR 2015*.

[24] D. P. Kingma and M. Welling (2014). Auto-encoding variational Bayes. *ICLR 2014*.

[25] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling (2016). Improved variational inference with inverse autoregressive flow. *NeurIPS 2016*.

[26] D. P. Kingma and P. Dhariwal (2018). Glow: Generative flow with invertible 1×1 convolutions. *NeurIPS 2018*.

[27] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le (2022). Flow matching for generative modeling. *ICLR 2023*.

[28] I. Loshchilov and F. Hutter (2019). Decoupled weight decay regularization. *ICLR 2019*.

[29] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu (2022). DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. *NeurIPS 2022*.

[30] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey (2015). Adversarial autoencoders. *ICLR 2016 Workshop*.

[31] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida (2018). Spectral normalization for generative adversarial networks. *ICLR 2018*.

[32] A. Q. Nichol and P. Dhariwal (2021). Improved denoising diffusion probabilistic models. *ICML 2021*.

[33] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis (2018). Parallel WaveNet: Fast high-fidelity speech synthesis. *ICML 2018*.

[34] G. Papamakarios, T. Pavlakou, and I. Murray (2017). Masked autoregressive flow for density estimation. *NeurIPS 2017*.

[35] W. Peebles and S. Xie (2023). Scalable diffusion models with transformers. *ICCV 2023*.

[36] A. Rahimi and B. Recht (2007). Random features for large-scale kernel machines. *NeurIPS 2007*.

[37] D. J. Rezende and S. Mohamed (2015). Variational inference with normalizing flows. *ICML 2015*.

[38] G. O. Roberts and J. S. Rosenthal (1998). Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society, Series B*, 60(1):255–268.

[39] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, R. Gontijo-Lopes, B. K. Ayan, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi (2022). Photorealistic text-to-image diffusion models with deep language understanding. *NeurIPS 2022*.

[40] Y. Song and S. Ermon (2019). Generative modeling by estimating gradients of the data

distribution. *NeurIPS 2019*.

[41] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole (2021). Score-based generative modeling through stochastic differential equations. *ICLR 2021*.

[42] J. Song, C. Meng, and S. Ermon (2021). Denoising diffusion implicit models. *ICLR 2021*.

[43] N. Tishby, F. C. Pereira, and W. Bialek (1999). The information bottleneck method. *Allerton Conference on Communication, Control, and Computing*.

[44] M. K. Titsias (2009). Variational learning of inducing variables in sparse Gaussian processes. *AISTATS 2009*.

[45] A. Tong, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, K. Fatras, G. Wolf, and Y. Bengio (2023). Improving and generalizing flow-matching for probability paths. *ICML 2023*.

[46] P. Vincent (2011). A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674.

[47] R. J. Williams (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

[48] A. G. Wilson, Z. Hu, R. R. Salakhutdinov, and E. P. Xing (2016). Deep kernel learning. *AISTATS 2016*.

[49] S. Zhao, J. Song, and S. Ermon (2019). InfoVAE: Balancing learning and inference in variational autoencoders. *AAAI 2019*.

[50] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena (2019). Self-attention generative adversarial networks. *ICML 2019*.